(1)

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

(1)

AFIT/GE/ENG/89D-51

IMPLEMENTATION OF ADAPTIVE ARRAYS IN THE BLOCK ORIENTED SYSTEMS SIMULATOR

THESIS

Joe H. Srubar
Captain, USAF

AFIT/GE/ENG/89D-51

Approved for public release; distribution unlimited

AFIT/GE/ENG/89D-51

IMPLEMENTATION OF ADAPTIVE ARRAYS IN THE BLOCK ORIENTED SYSTEMS SIMULATOR

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

Joe H. Srubar, B.S.
Captain, USAF

December, 1989

Acknowledgments

When trying to single out the people who helped the most during this thesis effort, the name of Lieutenant Colonel David Norman comes immediately to mind. As my thesis advisor, his suggestions, prompting, and help were invaluable. Mr. Dan Zambon was priceless in his immediate assistance whenever an obstacle had to be overcome on the computer systems. Captain Robert Williams provided some timely hints when certain problems were encountered. In the early part of my work, Major Glenn Prescott gave necessary insight to digital simulations at a crucial time.

No acknowledgement would be complete without a giant thank you to my wife Gail and daughters Betty and Susan. At times, the sacrifices they made far outweighed whatever I was going through. Without them, my work and life would have little meaning or satisfaction. What they have done during this time will not soon be forgotten.

Joe H. Srubar

Table of Contents

List of Figures

List of Tables

Abstract

This study investigated the feasability of implementing adaptive array algorithms in a simulation program called the Block Oriented Systems Simulator (BOSS). Two algorithms in particular were used: the least mean square (LMS) algorithm in real and complex form originally developed by Widrow, et. al. and the Applebaum algorithm. Two arrays were simulated with simple isotropic radiators used as the array elements. The first array used two elements spaced one half wavelength apart. The second array used four elements with variable geometry. The approach taken was to first implement the algorithms and arrays in the simulation language. Testing the algorithms with simple inputs and checking the algorithms ability to track these inputs was the second step. The last step was to test the algorithms with the arrays.

The results of the simulations showed the LMS algorithms implemented in BOSS were able to track DC signals along with sinusoidal inputs. The BOSS LMS implementation was also able to reject jammer signals by changing the gain and phase of the array elements. The amount of signal to jammer ratio was only a function of the gain limits put on the LMS loops. An analog implementation of the Applebaum algorithm did not perform as well. When used in a sidelobe canceller circuit with one adaptive weight, the complex weight would initially converge to a theoretical optimum weight but eventually would diverge. This problem was corrected with a discrete implementation of the algorithm. The discrete implementation provided weight convergence with nulls being placed at a jammer's spatial location. The null depth exactly matched the jammer power.

The simulations from this study show that adaptive array algorithms can be implemented and simulated with BOSS. The BOSS simulation environment proved to be an excellent tool for prototyping systems along with a providing a rich selection of post simulation output products.

IMPLEMENTATION OF ADAPTIVE ARRAYS IN THE BLOCK ORIENTED SYSTEMS SIMULATOR

I. Introduction

1.1 Background

The Information Transmission Branch of the Wright Research and Development Center wants to have a tool to evaluate contractors' proposals concerning adaptive arrays. This tool could be the traditional component level simulation which is difficult to modify and is generally applicable to only one array configuration. Or this tool could follow another approach in which components are implemented in a block fashion and simulated waveforms are processed by the blocks. If the latter approach to adaptive arrays proved feasible, a tool would be available which could be used for array evaluation and could be modified by simply moving blocks around the circuit diagram. Evaluation time could be reduced dramatically with designers required to have only a limited knowledge of computers and simulation.

Numerous studies have been performed on adaptive antenna arrays (2:1-271). By implementing different array patterns or spacings with different adaptive algorithms, various types of performance are achieved. In almost every case, some type of simulation was performed to predict the performance of the array and the adaptation algorithm. A common approach to the simulation is to model the array and algorithm in some high order language such as FORTRAN. This approach requires each element in the adaptive antenna be broken down into its smallest mathematical entity. These entities are then assembled and the simulation built from the bottom up, going from the model of the simplest resistor to the array and its environment. This type of simulation could be performed for any type of system. For instance,

a first order low pass filter could be modeled with a resistor and capacitor network similar to Figure (1).

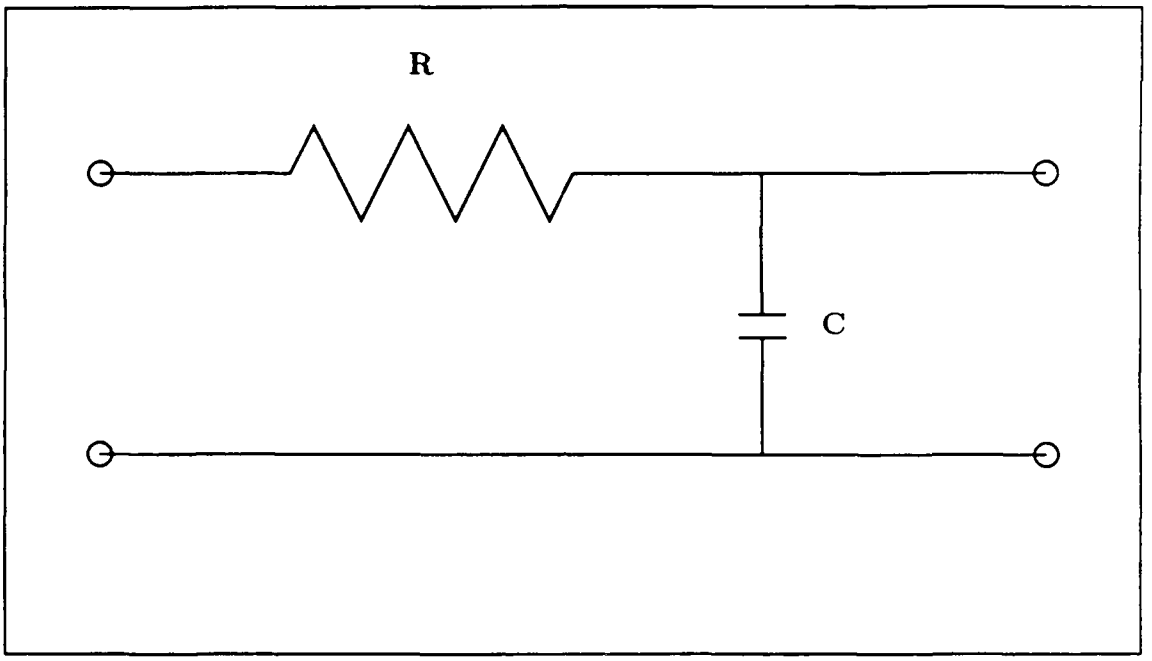

Figure 1. RC Low Pass Filter

The magnitude of the frequency response of this circuit is governed by the equation:

$$|H(s)| = \frac{1/RC}{\sqrt{(2\pi f)^2 + (1/RC)^2}} \tag{1}$$

where R is the resistor value, C the capacitor value and f the frequency. In a FORTRAN simulation of the frequency response, the code to perform this simulation might look like:

```
      DO 10 F = 0,500
        H(F)=(1/RC)/(SQRT((2*PI*F)**2+(R*C)**2))
   10 CONTINUE
```

Looking at the code does not give a feel for the circuit. Nor could this code be readily added to another simulation as a low pass filter unless the programmer had a thorough understanding of just what was implemented by this section of code.

However, there is a second approach to system simulation. With a software system such as the Block Oriented Systems Simulator (BOSS), the computer can be used to simulate waveforms that flow through the system (3:1-1). Using the simulated waveform approach differs from the original element simulation approach in which the computer evaluates complex formulas by replacing numbers in the formulas. In the RC circuit example, the computer replaces the frequency value F in the equation to simulate the circuit's frequency response.

The BOSS environment utilizes a set of templates for most elements used by communications engineers. The designer recalls these templates, fills in the necessary parameters, and builds a circuit or system from the "block" level rather than the component level. The designer can then evaluate the performance of the design from within the BOSS environment through built-in simulation tools. BOSS also provides many display options for simulation results through a post processor function. The BOSS graphics software that is presented to the user is written in LISP. BOSS then translates the user-entered block diagram into a FORTRAN simulation program (3:1-3). Primitive modules not implemented in BOSS can be built in FORTRAN and then imported as a BOSS template. Because BOSS was originally written to simulate communication systems, this feature allows a system designer to implement almost any circuit desired. The same RC filter in Figure (1) could be implemented in BOSS using a template available for low pass filters.

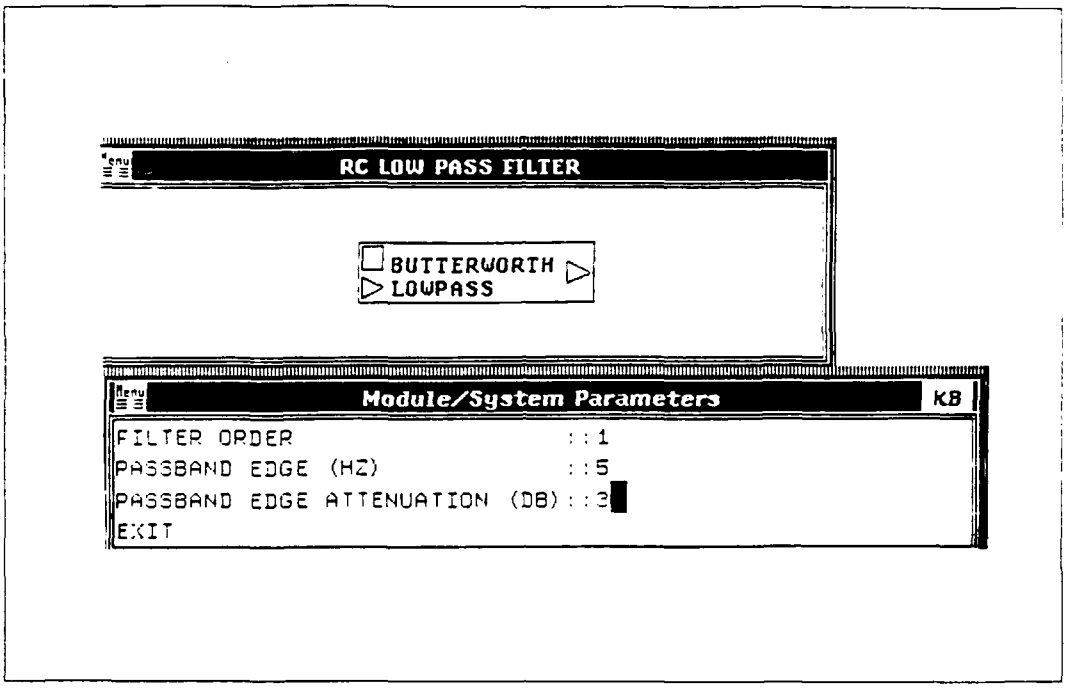

Figure 2. BOSS Low Pass Filter

Figure (2) shows a BOSS implementation of a first order low pass filter. Frequency response is one of the post processor functions available in BOSS and it is readily available to the user. Additionally, a designer can now add this low pass filter to other circuit components to build a system. The representation in Figure (2) gives the designer a feel for just what this component will do. This is the concept in BOSS. Systems are built from the block level.

An adaptive array might also be modeled from the block level rather than a component level in order to perform a simulation. One of the most common algorithms for adaptation is the Least Mean Square (LMS) algorithm. It can be broken down into integrators, summers, amplifiers, and time delays. The antenna elements themselves might be modeled as amplifiers with phase shifts dependent on their relative spacing. This approach could lead to modeling adaptive arrays in a block fashion.

1.2 Problem Statement

This thesis will investigate implementing adaptive antenna arrays within the Block Oriented Systems Simulator (BOSS). Specifically, two different adaptive algorithms along with two different arrays will be investigated to determine their suitability for block type simulation. If the algorithms and the arrays can be implemented in BOSS, results from simulations in this system can be compared with results of theoretical studies in the literature.

1.3 Summary of Current Knowledge

Since the late 1960s and early 1970s, a number of techniques for controlling adaptive antenna arrays have been developed. The concept of the adaptive array is to adjust the array weights in some way to enhance signal reception. This generally means enhancing the desired signal while reducing the effect of noise through a weight adjustment algorithm. The noise can be an interference source such as a jammer or simply thermal noise.

The most significant work on adaptive array algorithms began with the publication of the Widrow, Mantey, Griffith and Goode development of the least mean squares (LMS) algorithm for adaptive antenna systems (4:6). Another technique developed in 1966 but not published in the open literature until 1976 was the Applebaum algorithm (1:585). A number of other techniques have been developed including the power inversion algorithm, direct matrix inversion, and the update covariance matrix inverse algorithm (2:37). In each algorithm, there is an attempt to enhance the desired signal while reducing the effects of noise.

In the LMS algorithm, the weights are updated according to the following recursive equation:

$$\mathbf{W}_{j+1} = \mathbf{W}_j + 2\mu e_j \mathbf{X}_j \tag{2}$$

where

$\mathbf{W}$ is the weight vector

j is the iteration number

e_j is the difference between the output and the desired signals

μ is a convergence factor

$\mathbf{X}_j$ is the input vector at iteration j

This very compact form is easily implemented in a digital computer. The LMS algorithm requires a desired signal. The presence of the desired signal in the circuit should remove the need to go through this process at all. The key here is that the desired signal does not have to be an exact replica of what the system is expecting. The desired signal has only to be correlated with the expected input and uncorrelated with the noise. The receiver expecting the signal could produce the desired signal. This replica signal might not be in phase with the input signal but should be correlated with the input. Compton presents several other techniques for producing the desired signal as a function of the input (4:396–431).

The LMS algorithm attempts to minimize the mean square error between a desired signal and the system output. When the weights have adapted to an optimum point, the desired signal is enhanced and the noise reduced. The overall result is improved signal–to–noise ratio. There are times when the desired signal is unknown. However, there is still a need to improve the signal to noise ratio by reducing the effects of interference sources. The Applebaum algorithm does not require the desired signal and can be used in this case (2:47).

Like the LMS algorithm, the Applebaum algorithm adjusts the weights of an adaptive array in an attempt to maximize the signal–to–noise ratio (4:47). The antenna pattern produced by the adaptation process places antenna nulls in the

direction of noise sources. The weights in the Applebaum system are updated according to the following equation (1:593):

$$w_k(t) = G[\bar{t}_k(t) - \int_0^t x_k(\tau) \sum_{i=1}^{n} w_i(\tau) x_i(\tau) d\tau] \tag{3}$$

where w_k and w_i are the ith and kth weights, x_i is the ith input signal, $\bar{t}_k$ is the signal modulation seen by the kth array element, and G is a arbitrary constant. See Figure (10).

The key to this approach is that there is no need for the desired signal. If the desired signal is missing, the algorithm still attempts to remove the undesired noise.

1.4 Assumptions

In order to obtain statistically valid estimates of performance measures during a simulation, a large number of samples should be taken (3:1-1). According to sampling theory, the sample rate must be at least twice the highest frequency of any signal or waveform used in the simulation. If operating at the passband of a normal communication system, the highest frequency could be in the gigahertz range. The sampling rate would have to be twice this gigahertz frequency and one second of samples would involve two billion samples. In most applications, there is not enough storage space on the simulation computer to store two billion samples. Overcoming this situation requires that systems be represented by equivalent baseband envelopes. The simulations performed as part of this thesis will adopt the baseband approach and any non-linearities that exist at the original passband will be modeled at the lower frequencies. Using the baseband approach limits the fidelity to true waveforms used during simulations. To truly simulate signals that have been modulated to higher frequencies, a passband simulation should be performed. For this thesis effort, using baseband waveforms will not restrict results in any way.

1.5 Scope

Two adaptive array algorithms were implemented in BOSS. The first is the LMS algorithm and the second is the Applebaum algorithm. Both algorithms were used to update weights in an adaptive array. Two arrays were implemented in BOSS. The first array were two isotropic radiators spaced half a wavelength apart. The output of this array were fed into the weights along with a desired signal and noise. The noise source produced additive white Gaussian noise with power that can be set within the simulation. The second array consisted of four isotropic radiators arranged in a square with sides half a wavelength long and in a linear array with elements spaced half a wavelength apart. Again, the output of the array were fed into a set of four weights controlled first by the LMS algorithm and then the Applebaum algorithm. The output of these arrays along with array directivity patterns were compared with other studies using the same arrays and algorithms to verify performance.

1.6 Approach/Methodology

The first step in approaching the problem is to determine if either of the adaptive algorithms can be implemented in BOSS. A single LMS loop was implemented in BOSS and used to update one weight. The weight was initially be set to zero and a sinusoidal waveform was used as both the input and desired signal. A simulation was performed at this point. The results of this simulation showed whether or not the LMS algorithm was adjusting the weight to track the incoming sinusoid. Additional weights were added along with phase shifters to provide 90 degrees of phase shift so that the weights could produce any phase angle. These additional weights and phase shifters were connected to the simulated two element array. A desired signal along with a noise source formed the inputs to the array.

Once the LMS algorithm proved successful, the Applebaum algorithm was implemented using the same approach. One weight was controlled by the algorithm

while tracking a sinusoid. More weights along with the array were added to verify this algorithm's performance.

The next step was implementing the four element array. This array was connected to a four weight system updated with one algorithm. Then the other algorithm was tested. Each step along the way, the BOSS results were compared with simulation results from previous studies and theoretical predictions.

1.7 Materials and Equipment

The BOSS software requires a Digital Equipment Corporation (DEC) VAX Station running the VMS operating system or a SUN-3 workstation running the UNIX operating system (3: 1-3). The DEC equipment is currently available in the Air Force Institute of Technology (AFIT) Signal Processing Laboratory in Building 642.

1.8 Summary

If the BOSS approach to modeling and simulating adaptive arrays proves feasible, a tool will be available which could be used for array evaluation and could be modified by simply moving blocks around the BOSS circuit diagram. Designers would require only a limited knowledge of simulation techniques. Evaluation of contractor's proposals would be simpler and faster.

This thesis is divided into five chapters. The first chapter introduced the problem along with an approach to solving the problem. Additionally, the first chapter covered the objectives of this research, some basic assumptions, the scope of the work to be done, and the equipment on which BOSS operates. The second chapter is a literature review covering work that has been done on the LMS and Applebaum algorithms as applied to adaptive arrays. The second chapter also gives further explanation of terms and concepts along with detailed development of the theory used. Chapter three covers the methodology used. This chapter tells what work was

done and how it was accomplished. Chapter four presents the findings. The fifth chapter draws conclusions based on chapter four findings and makes recommendations for further research.

II. Adaptive Array Theory

2.1 The Least Mean Square Algorithm

As mentioned in chapter 1, the Least Mean Square (LMS) algorithm was introduced by Widrow et al. in 1967 (10:2143). The development of this algorithm as used by this thesis will follow the development in the original paper. Figure (3) represents a continuous time basic adaptive array. The purpose of the weights

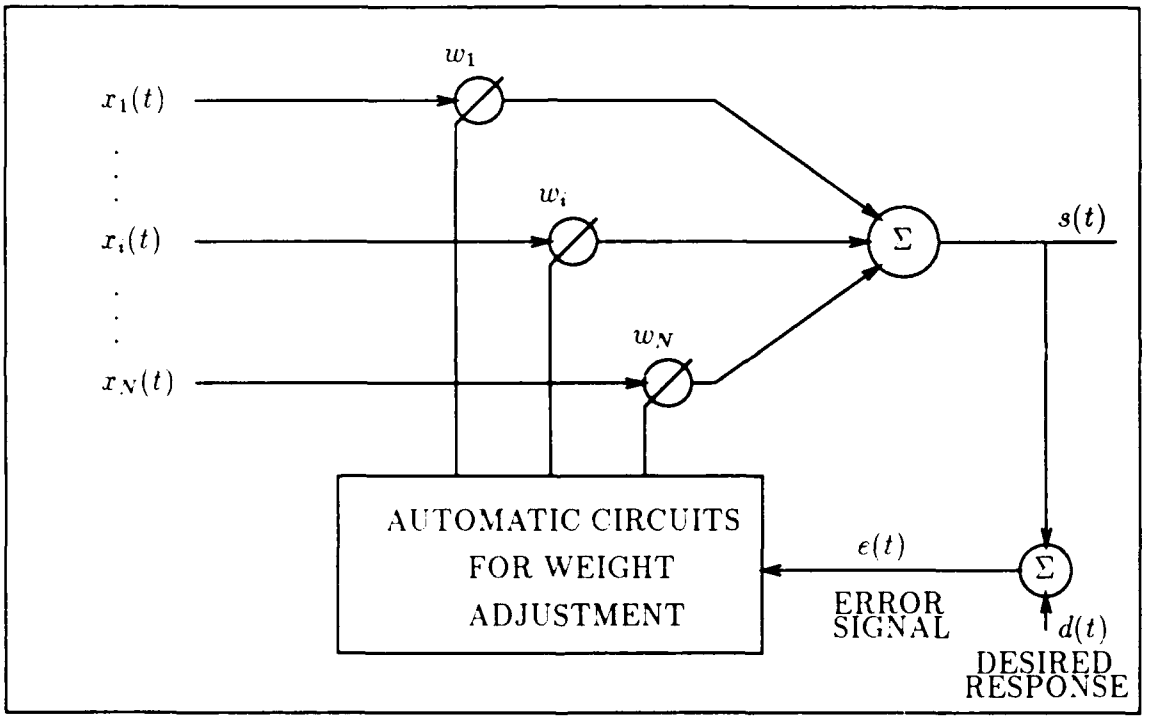

Figure 3. Basic Adaptive Array

$w_1, w_2, \cdots, w_N$ is to reduce the mean square error between the desired signal and the output of the array. The output of the array, $s(t)$, is formed by multiplying the inputs to the array, $x_1(t), x_2(t), \cdots, x_N(t)$ by the corresponding weights.

$$s(t) = \sum_{i=1}^{N} x_i(t) w_i \tag{4}$$

where N is the number of weights. The output may be alternatively expressed in vector form

$$s(t) = \mathbf{W^t X}(t) \tag{5}$$

where $\mathbf{W^t}$ is the vector

$$\mathbf{W^t} = [w_1 \; w_2 \; \cdots \; w_N] \tag{6}$$

and $\mathbf{X}(t)$ is the input vector

$$\mathbf{X}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_N(t) \end{bmatrix} \tag{7}$$

In a simulation where the signals are sampled at discrete time steps, the array output is described by

$$s_j = \mathbf{W^t X}_j \tag{8}$$

where j is the jth discrete sample. The array output s_j is now compared with a desired signal d_j. Subtracting the array output from the desired signal produces an error at each discrete sample point described by

$$e_j = d_j - \mathbf{W^t X}_j \tag{9}$$

where e_j is the error at each jth sample point and d_j is the desired signal at each jth sample point. Squaring both sides of Eq (9) and then taking the expected value of both sides gives

$$E[\, e_j^2 \,] = E[\, d_j^2 - 2\mathbf{W^t} d_j \mathbf{X}_j + \mathbf{W^t X}_j \mathbf{X^t}_j \mathbf{W} \,] \tag{10}$$

where $E[\,]$ represents the expectation operator. Letting $\mathbf{R}$ represent the matrix of autocorrelation and cross correlation elements of $E[\, \mathbf{X}_j \mathbf{X^t}_j \,]$ and $\mathbf{P}$ represent the

vector of cross correlation elements of $E[\ \mathbf{X}_j d_j\ ]$ simplifies Eq (10) to

$$E[\ e_j^2\ ] = E[\ d_j^2\ ] - 2\mathbf{W}^t\mathbf{P} + \mathbf{W}^t\mathbf{R}\mathbf{W} \tag{11}$$

Taking the derivative of Eq (11) with respect to each one of the weights yields the gradient of a quadratic surface that represents the weight plane.

$$\nabla E[\ e^2\ ] = 2\mathbf{R}\mathbf{W} - 2\mathbf{P} \tag{12}$$

where ∇ is the gradient operator. Setting the results of Eq (12) to zero produces a set of linear simultaneous equations that can be solved for the optimum weight values.

$$\mathbf{W}^* = \mathbf{R}^{-1}\ \mathbf{P} \tag{13}$$

where $\mathbf{W}^*$ is the optimum weight vector. When the optimum weight values in Eq (13) are used in Eq (8), the least mean square error between the array output and the desired signal is produced (10:2143-2148).

Looking at Eqs (12) and (13) two problems arise when trying to develop what might be considered a real time adaptive array system. The first problem is calculating the input statistics. To truly develop the auto and cross-correlation products of the input signal along with the cross-correlation product between the desired and input signals, a large number of samples need to be taken and then multiplied and summed to form a valid sample size. By the time these products are formed, the signal statistics can change because the relative position of the inputs to the array have moved while the large number of samples are collected. Even in modern computer systems the muliplications and additions take time. The second problem occurs in Eq (13). Here, the inverse of the $\mathbf{R}$ matrix has to be computed. A matrix inversion can take up to $n^5/3$ multiplications alone where n is the number of weights (7:30-31). Clearly, there must be an approximation made to the exact solution for

the optimum weight vector in Eq (13). This is the next step in developing a useful real time weight update algorithm.

Figure (4) shows a portion of a two-dimensional performance surface for a system with two weights (9:21). The vertical axis of Figure (4) represents the mean

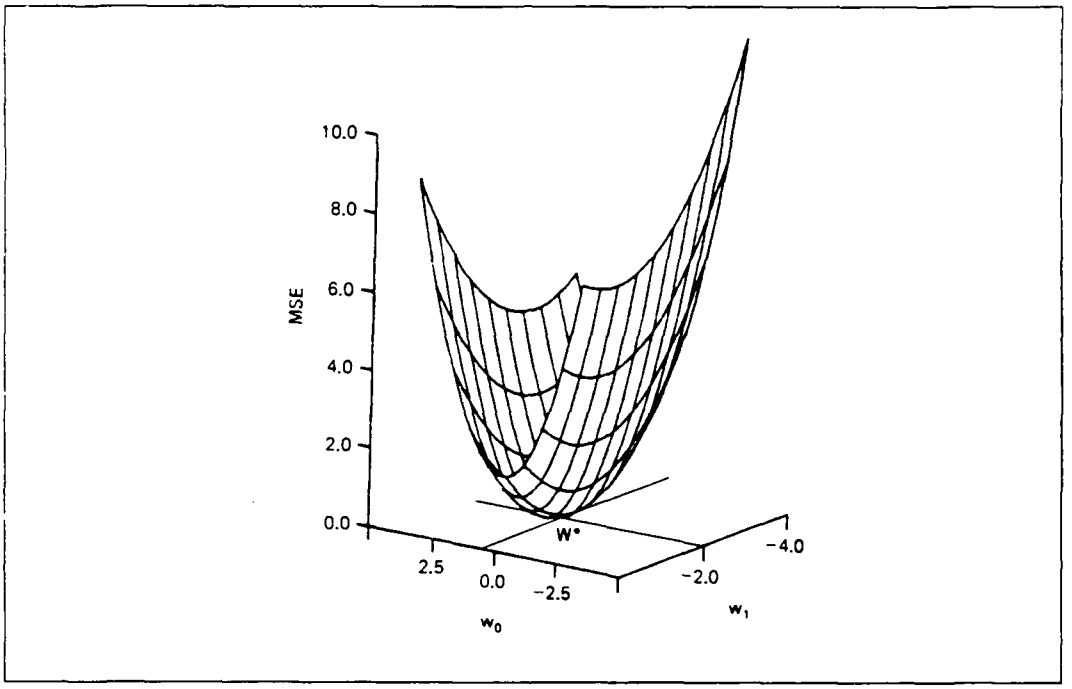

Figure 4. Two-Dimensional Performance Surface (9:21)

square error as a function of two weights. Since the surface is quadratic in nature, only a single global minimum exits. The LMS algorithm searchs the gradient of this surface for the global minimum. The values of the weights are updated in the direction of the gradient. The weight values for the next iteration in the simulation are given by

$$\mathbf{W}_{j+1} = \mathbf{W}_j - \mu\widehat{\nabla}_j \tag{14}$$

where $\mathbf{W}_{j+1}$ is the weight vector for the next simulation iteration, $\mathbf{W}_j$ is the weight vector for the current iteration, μ is a scalar convergence constant, and $\widehat{\nabla}_j$ is the

gradient vector estimate of the squared error with respect to $\mathbf{W}$ and is given by

$$\widehat{\nabla}_j = \nabla[\ e_j^2\ ] = 2e_j[\nabla[\ e_j\ ]] \tag{15}$$

Combining Eqs (14), (15), and (9) gives

$$\nabla[\ e_j\ ] = \nabla[d_j - \mathbf{W}^t{}_j\mathbf{X}_j\ ] \tag{16}$$

Since Eq (16) is a single sample at one instant in time, the gradient is a function of the weights and not the desired signal and is given by

$$\nabla[\ e_j\ ] = -\mathbf{X}_j \tag{17}$$

Now Eq (17) can be substituted into Eq (15).

$$\widehat{\nabla}_j = -2e_j\mathbf{X}_j \tag{18}$$

Using the gradient estimate in Eq (18) in Eq (14) yields the LMS weight update equation

$$\mathbf{W}_{j+1} = \mathbf{W}_j + 2\mu e_j\mathbf{X}_j \tag{19}$$

Equation (19) can be written as a differential equation which has a solution in the form

$$\mathbf{W}(t) = 2\mu \int_0^t e(\tau)\mathbf{X}(\tau)\ d\tau \tag{20}$$

This continuous time representation of Eq (19) can be implemented in an analog circuit. Figure (5) shows an analog circuit that implements Eq (20) while Figure (6) shows a digital realization of Eq (19) (10:2149).

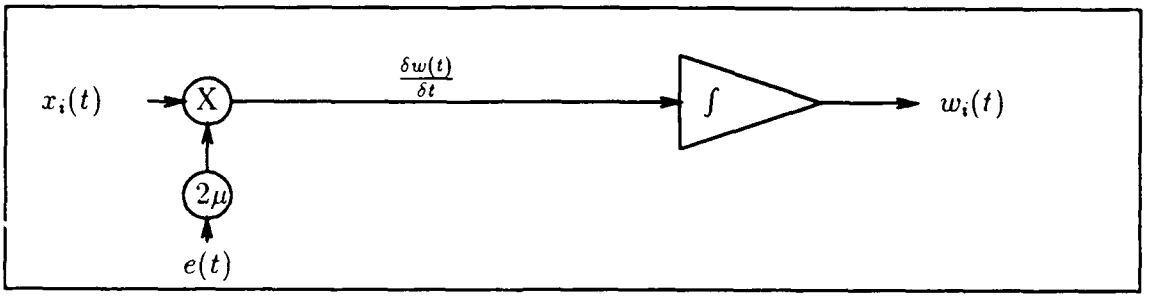

Figure 5. Analog LMS Loop (10:2149)

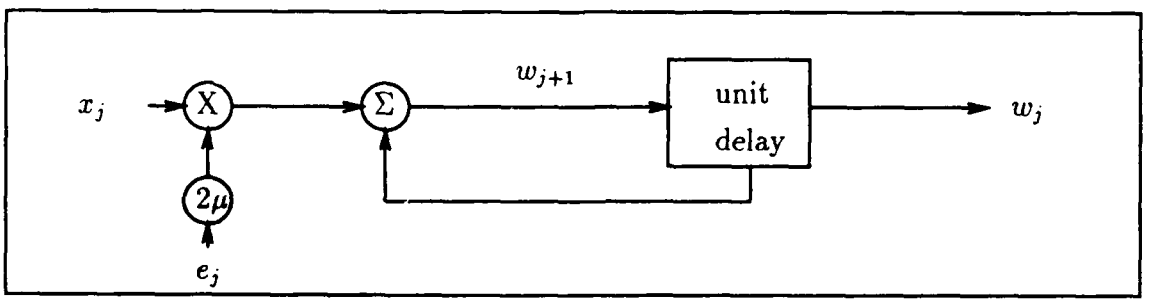

Figure 6. Digital LMS Loop (10:2149)

2.2 *Complex LMS Algorithm*

The LMS algorithm was expanded to include a complex derivation in 1975 (11:719). Figure (7) shows a block diagram of a complex LMS adaptive array (4:7). The inputs and the weights from Eqs (7) and (6) are now complex and represented

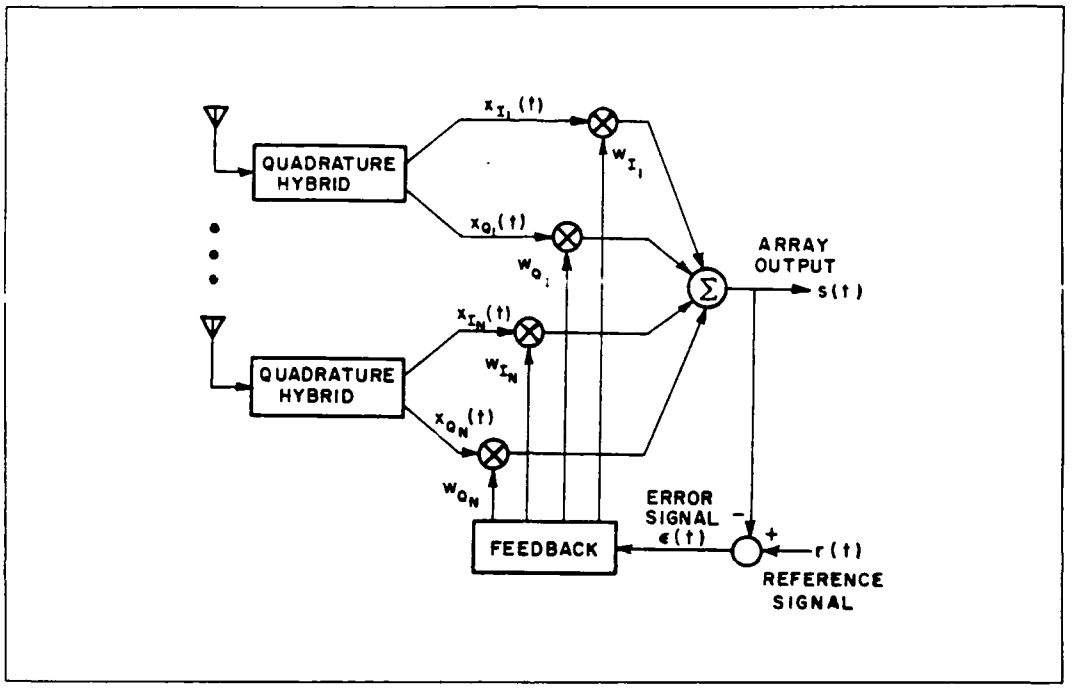

Figure 7. Complex Adaptive Array (4:7)

by

$$\mathbf{X}_j = \begin{bmatrix} x_{1Rj} \\ x_{2Rj} \\ \vdots \\ x_{nRj} \end{bmatrix} + i \begin{bmatrix} x_{1Ij} \\ x_{2Ij} \\ \vdots \\ x_{nIj} \end{bmatrix} = \mathbf{X}_{Rj} + i\mathbf{X}_{Ij} \tag{21}$$

and

$$\mathbf{W}_j = \begin{bmatrix} w_{1Rj} \\ w_{2Rj} \\ \vdots \\ w_{nRj} \end{bmatrix} + i \begin{bmatrix} w_{1Ij} \\ w_{2Ij} \\ \vdots \\ w_{nIj} \end{bmatrix} = \mathbf{W}_{Rj} + i\mathbf{W}_{Ij} \tag{22}$$

where R designates the real component and I designates the imaginary component. The error, desired, and output signals are also complex and given by

$$e_j = e_{Rj} + ie_{Ij} \tag{23}$$

$$d_j = d_{Rj} + id_{Ij} \tag{24}$$

$$y_j = y_{Rj} + iy_{Ij} \tag{25}$$

The output and error signals are again formed in accordance with Eqs (8) and (9) but the complex forms of Eqs (21) through (25) are used.

$$y_j = \mathbf{X}^t{}_j \mathbf{W}_j = \mathbf{W}^t{}_j \mathbf{X}_j \tag{26}$$

$$e_j = d_j - y_j = d_j - \mathbf{X}^t{}_j \mathbf{W}_j \tag{27}$$

All additions and multiplications obey the rules of complex algebra.

The concept is again to minimize the mean square error. This is accomplished by adjusting both the real and imaginary parts of the complex weights to minimize the real and complex parts of the error signal in Eq (27). In the complex case, reduction of the mean square error is replaced by reduction of the average total error power. Assuming the real and imaginary components of the error power are uncorrelated, the error power is given by

$$E[\, e_j \bar{e}_j \,] = E[\, e_{Rj}^2 \,] + E[\, e_{Ij}^2 \,] \tag{28}$$

where the bar above $\bar{e}_j$ designates the complex conjugate and is given by

$$\bar{e}_j = \bar{d}_j - \bar{\mathbf{X}}_j^t \bar{\mathbf{W}}_j \tag{29}$$

Again, the gradient of the error squared must be taken but this time with respect

to the real and imaginary components of the weight vector. The resulting equations are

$$\nabla_R(e_j\bar{e}_j) = e_j[\nabla_R(\bar{e}_j)] + \bar{e}_j[\nabla_R(e_j)] = e_j(-\bar{\mathbf{X}}_j) + \bar{e}_j(-\mathbf{X}_j) \tag{30}$$

and

$$\nabla_I(e_j\bar{e}_j) = e_j[\nabla_I(\bar{e}_j)] + \bar{e}_j[\nabla_I(e_j)] = e_j(i\bar{\mathbf{X}}_j) + \bar{e}_j(-i\mathbf{X}_j) \tag{31}$$

An estimate of the gradient is again made and the weight update equation for the real and imaginary components becomes the complex form of the LMS algorithm given by (11:719-720)

$$\mathbf{W}_{j+1} = \mathbf{W}_j + 2\mu e_j\bar{\mathbf{X}}_j \tag{32}$$

Both the real and imaginary components of Eq (32) can be implemented in loops similar to Figures (5) and (6). Figure (8) shows an analog implementation of a complex LMS loop while Figure (9) shows a discrete implementation where the $1/S$ component represents an integrator (4:22).

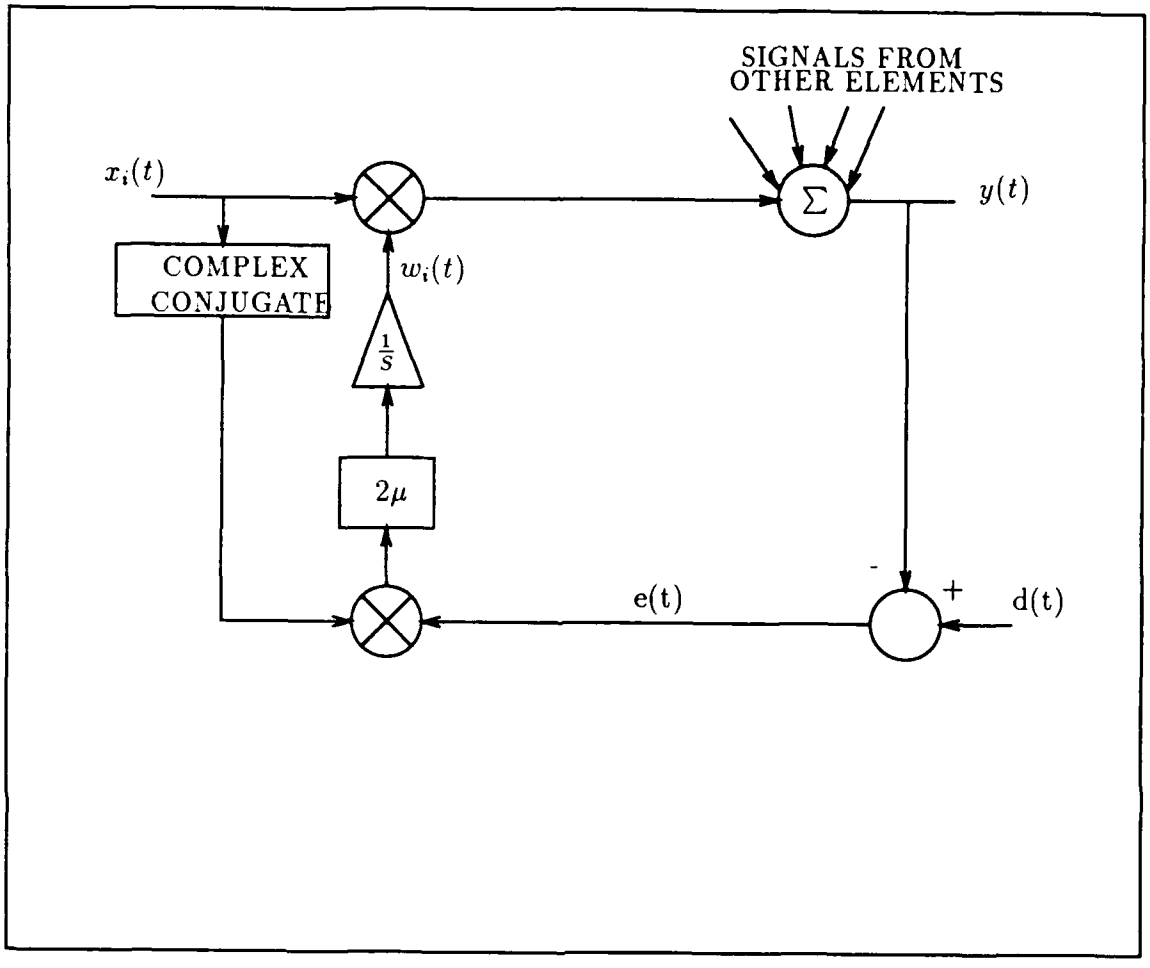

Figure 8. Complex Analog LMS Loop (4:22)

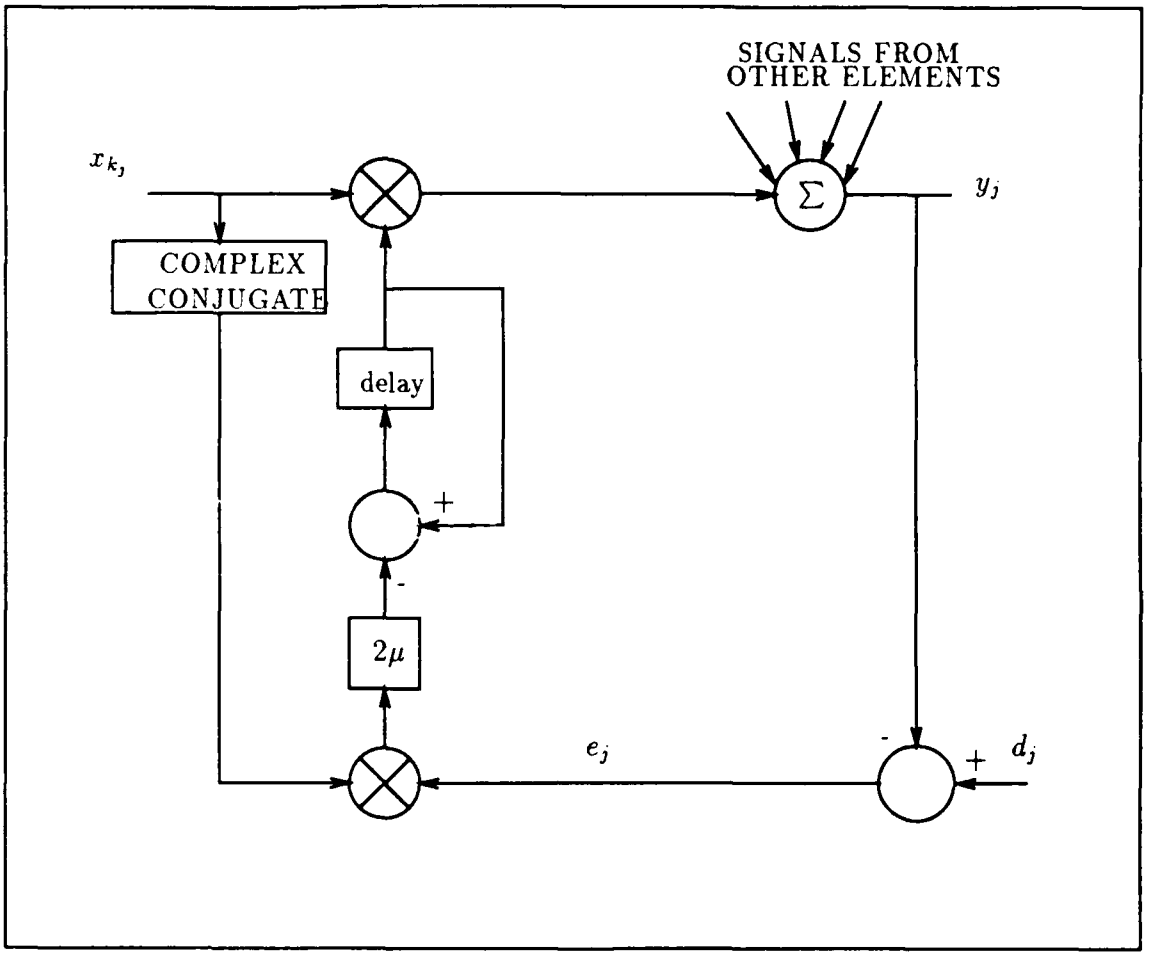

Figure 9. Complex Discrete LMS Loop

2.3 The Applebaum Algorithm

Rather than attempting to reduce the mean square error between the output of the array and some desired signal, the Applebaum algorithm attempts to maximize the signal to noise ratio by adjusting the weights in the adaptive linear combiner of Figure (7) (1:585). The development of this algorithm starts by breaking the noise and signal parts into separate components for evaluation. This development follows Applebaum's original work and uses the notation from that work. Using an approach

similar to Eqs (4) and (5), the output of the array due to the signal is given by

$$v_s = \alpha \sum_{k=1}^{K} x_k \;\; w_k \tag{33}$$

where v_s is the output due to the signal, x_k and w_k are given by Eqs (7) and (6), and α defines the amplitude and time variation of the signal. Again, the output can be expressed in matrix form by

$$v_s = \alpha \mathbf{W}^t \mathbf{X} = \alpha \mathbf{X}^t \mathbf{W} \tag{34}$$

The noise component of the output is given by

$$v_n = \mathbf{W}^t \mathbf{N} = \mathbf{N}^t \mathbf{W} \tag{35}$$

where $\mathbf{N}$ is

$$\mathbf{N} = \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_K \end{bmatrix} \tag{36}$$

The expected output noise power is given by

$$P_n = E[\, |v_n|^2 \,] = \bar{\mathbf{W}}^t E[\, \bar{\mathbf{N}} \mathbf{N}^t \,] \mathbf{W} \tag{37}$$

$$= \bar{\mathbf{W}}^t \mathbf{M} \mathbf{W} \tag{38}$$

where

$$\mathbf{M} = E[\, \bar{\mathbf{N}} \mathbf{N}^t \,] = [\, \mu_{kl} \,] \tag{39}$$

where μ_{kl} is the kth element of row l. Note that the P_n is not the same entity as P in the LMS development.

The $\mathbf{M}$ matrix is Hermitian since $\bar{\mathbf{M}}^t = \mathbf{M}$. $\mathbf{M}$ is also positive definite because

the output noise power, P_n is greater than zero whenever $\mathbf{W} \neq 0$ (1:586). These two facts lead to a transformation matrix where all channels have uncorrelated noise of equal power. Using this concept, and using $\mathbf{A}$ as the transformation matrix, Eqs (34) and (35) become

$$v_s = \alpha \widehat{\mathbf{W}}^t \widehat{\mathbf{X}} \tag{40}$$

$$v_n = \widehat{\mathbf{W}}^t \widehat{\mathbf{N}} \tag{41}$$

where

$$\widehat{\mathbf{X}} = \mathbf{A}\mathbf{X} \tag{42}$$

$$\widehat{\mathbf{N}} = \mathbf{A}\mathbf{N} \tag{43}$$

$$\mathbf{W} = \mathbf{A}^t \widehat{\mathbf{W}} \tag{44}$$

The $\mathbf{A}$ matrix equalizes and decorrelates the noise. and

$$E[\, \bar{\mathbf{A}}\bar{\mathbf{N}}\widehat{\mathbf{N}}^t \,] = \mathbf{I} \tag{45}$$

where $\mathbf{I}$ is a k by k identity matrix. With this result, the noise power simply becomes the magnitude of $\widehat{\mathbf{W}}$ squared. Again using Eq (45), The noise power can be rewritten to

$$P_n = \widehat{\mathbf{W}}^t \mathbf{M} \widehat{\mathbf{W}} \tag{46}$$

The result of the last three equations is that a relationship can be developed between the $\mathbf{M}$ and $\mathbf{A}$ matrices. This relationship is described by

$$\bar{\mathbf{A}}\mathbf{M}\mathbf{A}^t = \mathbf{I} \tag{47}$$

$$\mathbf{M} = (\mathbf{A}^t \bar{\mathbf{A}})^{-1} \tag{48}$$

As Applebaum points out, the optimum weight vector occurs when the noise com-

ponents in each channel of the adaptive array have equalized, uncorrelated powers. Now the optimum weight vector can be written as

$$\mathbf{W}^* = \mathbf{A}^t\widehat{\mathbf{W}}^* = \mathbf{A}^t\mu\bar{\mathbf{A}}\bar{\mathbf{X}} \tag{49}$$

$$= \mu\hat{\mathbf{M}}^{-1}\bar{\mathbf{X}} \tag{50}$$

where μ is an arbitrary constant. Whenever Eq (50) is met, the maximum signal to noise ratio is achieved (1:586–587). A weight update algorithm that will implement Eq (50) is provided below.

$$w_k(t) = G[\bar{t}_k(t) - \int_0^t x_k(\tau)\sum_{i=1}^{n} w_i(\tau)x_i(\tau)d\tau] \tag{51}$$

where $w_k(t)$ is the weight value for the kth channel, $\bar{t}_k(t)$ is the desired vector component for the kth channel, G is an amplification factor, and $x_k(t)$ is the input to the kth channel.

A practical implementation of the development above must exist in order to provide control of the adaptive array. The functional block diagram of a circuit implementing Eq (51) is shown is Figure (10) (1:593).

2.4 *Summary*

Figures (5), (6), and (10) form the basis for update circuits to be used in BOSS. In each case, the algorithms lend themselves to a block type implementation. This fact leads to the approach to be taken toward implementing adaptive arrays in BOSS and is the subject of the next chapter.

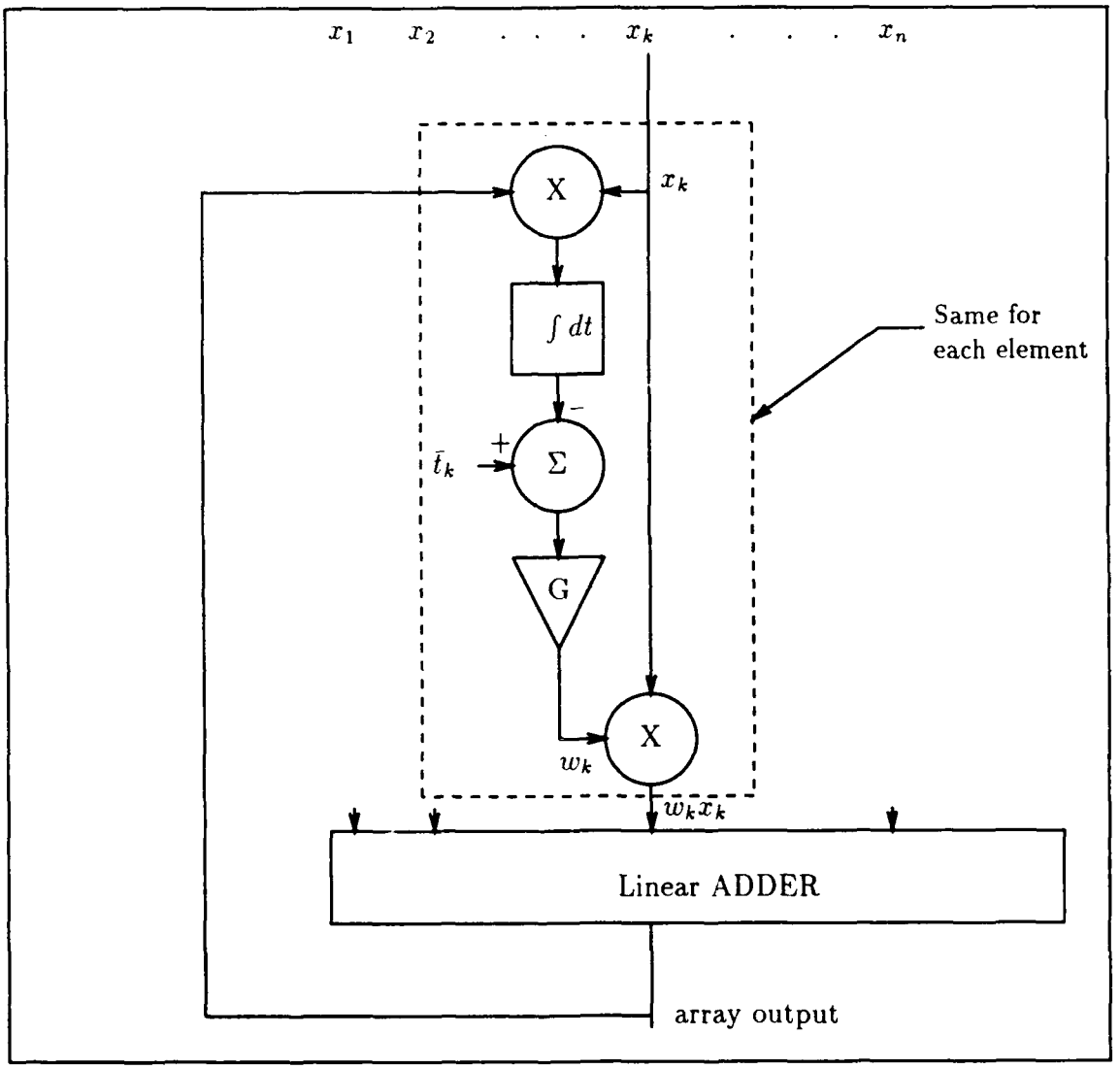

Figure 10. Applebaum Weight Update Block Diagram (1:593)

III. Implementation Approach

3.1 Implementing the Algorithms

Chapter 2 provided the mathematical basis for the LMS, the complex LMS and the Applebaum algorithms for updating weights in an adaptive array. Equation (19) gives a weight update formula based on the LMS algorithm. The algorithm contained in the equation was then translated into a block diagram in Figure (6). The analog form of this equation is given in Eq (20) with a block diagram in Figure (5). The complex LMS and Applebaum algorithms also have equations and block diagrams in Chapter 2. This chapter will show the approach to implementing those algorithms in a block diagram form in BOSS. The approach to implementing both the two and four element arrays will be covered. These arrays will be implemented to handle both real and complex signals. The first step is to implement the LMS algorithm.

3.2 LMS Algorithm Implementation

The components in Figure (5) are two multipliers, an integrator, and a gain factor of 2μ. BOSS provides two and three input multpliers in the arithmetic section of its *Basic Building Blocks*. The gain coefficient is provided in the *Basic Building Blocks* as a module called *REAL COEFFICIENT GAIN*. BOSS also has an integrator in its filter section. Putting these modules together yields an analog LMS loop. The BOSS implementation is shown in Figure (11).

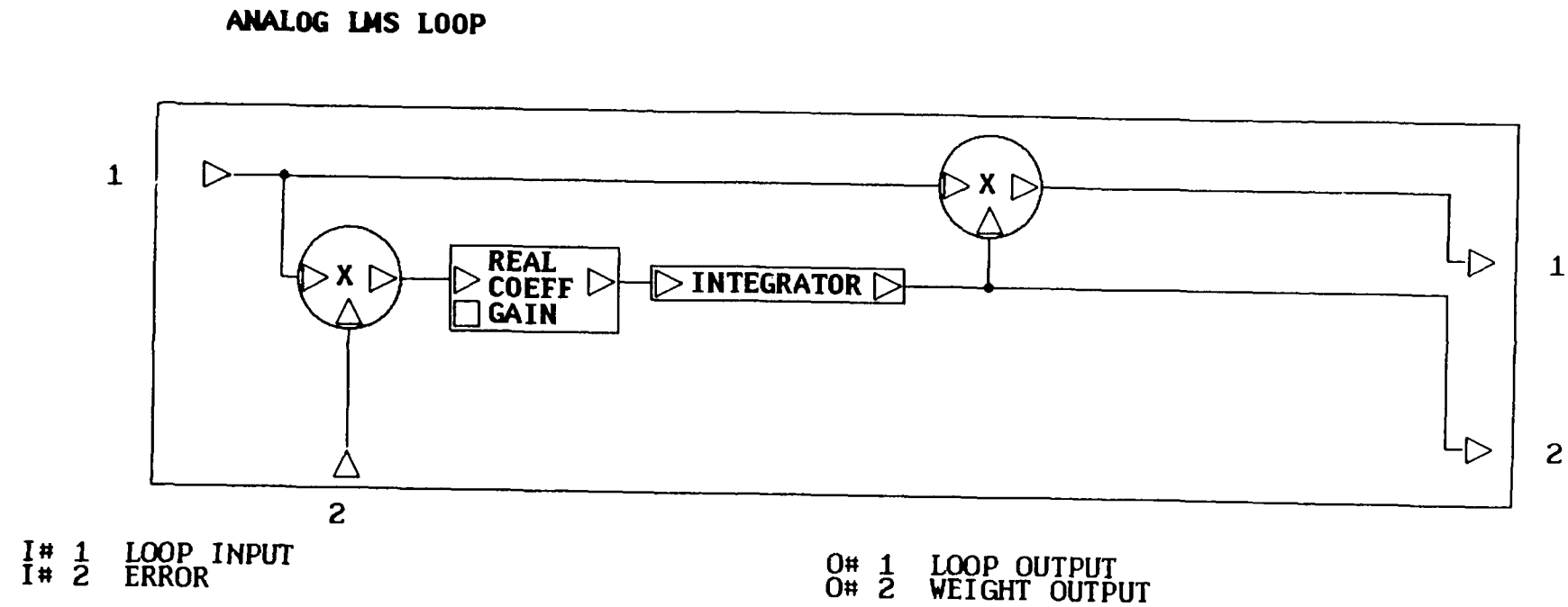

Figure 11. BOSS Analog LMS Loop

The block diagram in Figure (11) was produced with the BOSS *Print Module Block Diagram* command. The two small triangles which point in at the bottom and left side of the diagram are inputs to the circuit and are labeled I# 1 and I# 2. Input 1 is the loop input from the array element feeding this particular LMS loop while input 2 is the error formed by the difference between the desired signal and the system output. The error signal will be formed at a higher level and fed back to this part of the circuit. The two outputs are labeled O# 1 and O# 2 and are the loop output and the weight value. The outputs are on the right side of the block diagram and are represented by triangles that point out of the circuit. Output 1 is simply $w_i(t)x_i(t)$. The weight value is output so that a time history of the weight's behavior is available. The *REAL COEFF GAIN* represents the 2μ convergence factor and is exported from this level of circuit abstraction as an adjustable parameter.

Using the same approach for the discrete implementation as for the analog implementation, the two multipliers and adder from the *Basic Building Blocks* were used. The unit delay in Figure (6) is available in BOSS's *Memory* section of the *Basic Building Blocks*. The same gain coefficient was used to provide the 2μ factor. The BOSS implementation is shown in Figure (12). The inputs and outputs to this loop are the same as in Figure (11). With both the discrete and analog forms of the LMS algorithm implemented, the next step is to implement the complex LMS algorithm.

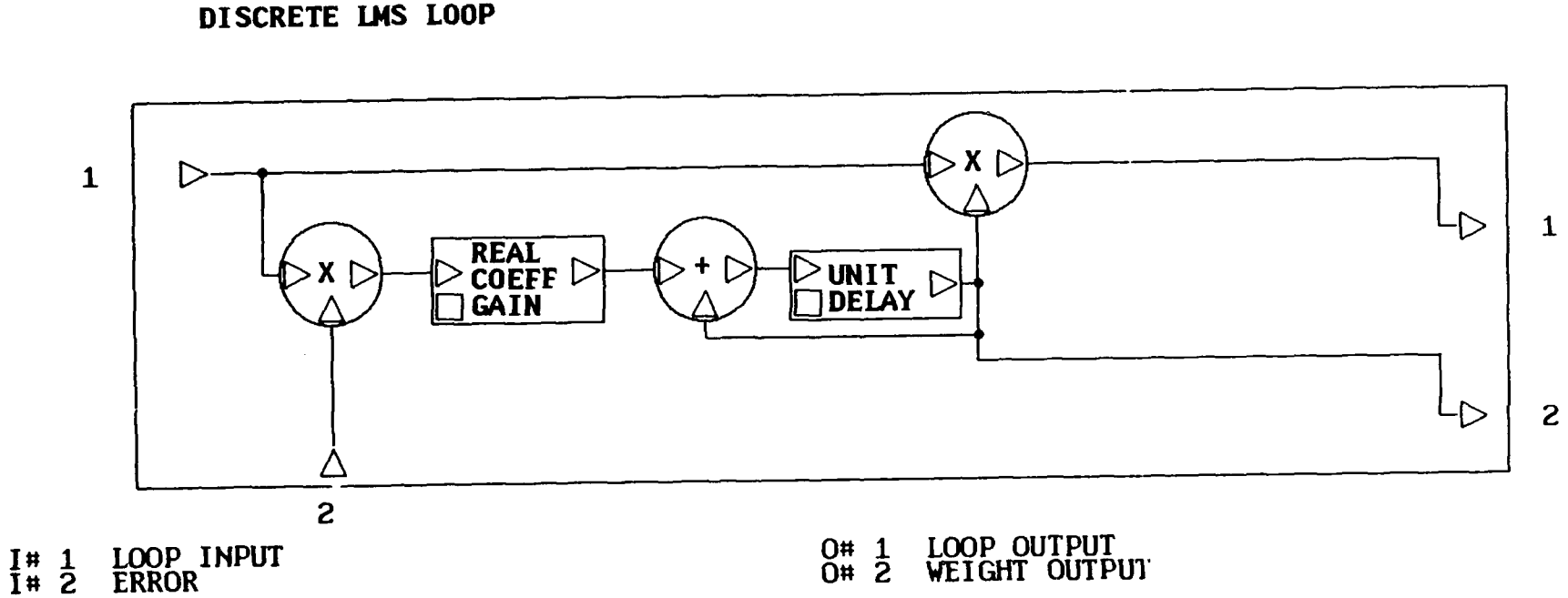

Figure 12. BOSS Discrete LMS Loop

3.3 Complex LMS Algorithm Implementation

Chapter 2 laid the groundwork for implementing the complex LMS algorithm in Eq (32) and Figures (8) and (9). The only new module necessary in this algorithm, not included in the previous algorithms, is one for obtaining the complex conjugate of a complex number. BOSS contains complex conjugate in its *Type/Unit Conversion* section within *Basic Building Blocks*. The real gain coefficient can no longer be used for the 2μ factor because the input to the real gain coefficient module must be a real value. However, BOSS contains an untyped gain module *Arithmetic* section of the *Basic Building Blocks*. The input to this module can be any type with BOSS making the type determination and multiplying according to real or complex algebra. Additionally, the adders, multipliers and unit delay modules have untyped inputs allowing either real or complex inputs and outputs. Using these BOSS modules, the complex LMS loop was implemented in Figure (13). The form of this loop is slightly different than the other LMS loops. Rather than forming the output by multiplying the input by the weight value, the output of this circuit is the weight value only. The output for the section must be formed by multiplying the weight by the input at a higher level. The input on the left side of the circuit is a complex signal from one of the array elements. The input at the bottom of the circuit is the error signal which is the difference between the desired and weighted array output signals.

3.4 Applebaum Algorithm Implementation

The Applebaum algorithm, like the LMS algorithm, is implemented in a block diagram with an adder, multipliers, and an itegrator. The block diagram is shown in Figure (10). BOSS provides all the necessary components to implement this alrorithm. The BOSS implementation is shown in Figure (14).

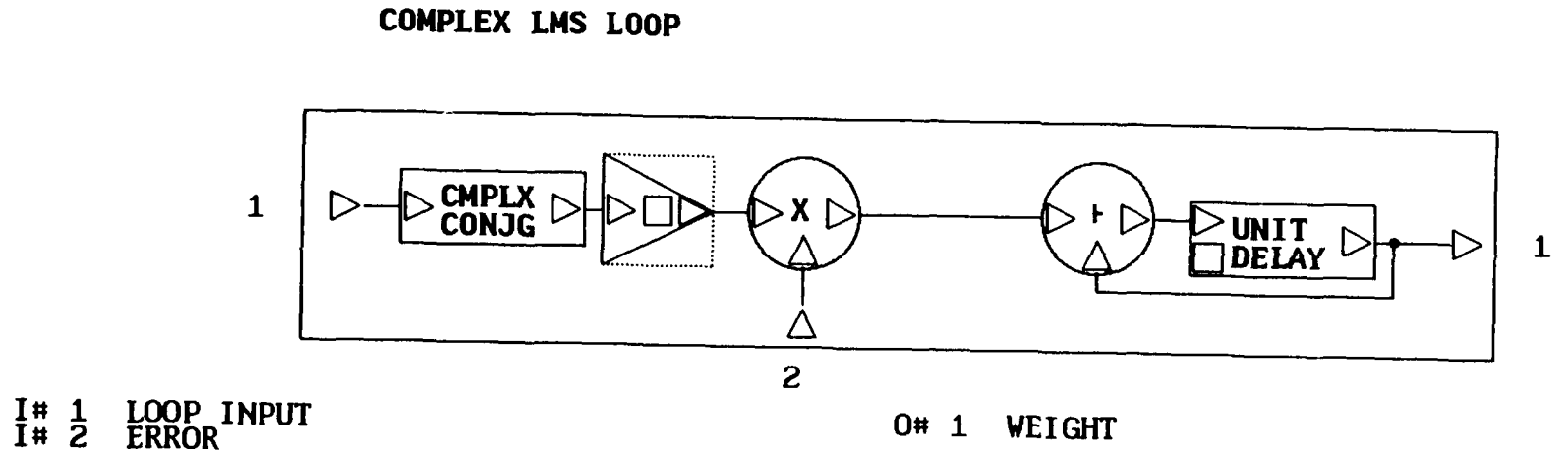

Figure 13. BOSS Complex LMS Loop

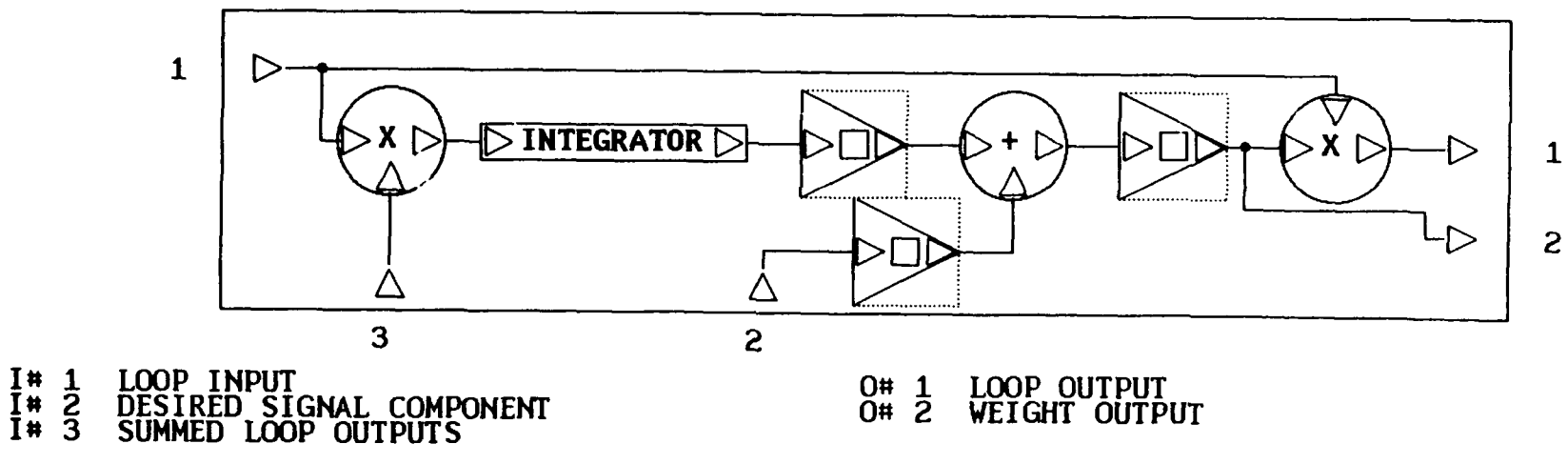

Figure 14. BOSS Applebaum Loop

3.5 *Array Implementation*

With the algorithms implemented, the next step is to implement the two and four element arrays. Figure (15) shows a two element array with a desired signal arriving at angle θ_d with reference to element 1. The two elements are spaced $\lambda/2$ apart where λ is the wavelength of the center frequency of the desired signal as well as the array wavelength. The desired signal will arrive at element 2 at a different time than at element 1. This causes a phase shift between the signals at the two elements. In the case of Figure (15), the phase shift is given by the equation (4:24)

$$\phi = \pi sin(\theta_d) \tag{52}$$

The difference in arrival time and phase shift for this array is applicable to any signal whether it is the desired, a jammer, or a noise source. To implement the array in Figure (15) using BOSS, the array was broken into three different parts. The first part processed the desired signal. If element 1 is used as a reference, the signal seen by element 1 should simply be passed through the array to the signal processing section. This is accomplished with a unity gain element. The signal received from element 2 should have a phase shift in accordance with Eq (52). For real signals, BOSS provides a phase parameter in its sinusoidal source only. However, the phase parameter could be provided as a time delay within the array implementation. To provide the correct phase shift, the incoming signal is delayed an appropriate number of time samples. BOSS provides a multi- sample delay in its *Memory* section of the *Basic Building Blocks.* The module is called *Multi-Stage Delay.* Using the delay element along with a constant gain module the signal section of the array was constructed within BOSS and is shown in Figure (16). Similarly, the jammer and noise sections were implemented and are shown in Figures (17) and (18).

The array itself is now formed by combining the three sections. The element 1 outputs of the signal, jammer, and noise sections are added together to form total

element 1 output of the array. The element 2 outputs from the three sections are added also added together to form the total element 2 output. The resulting array is show in Figure (19).

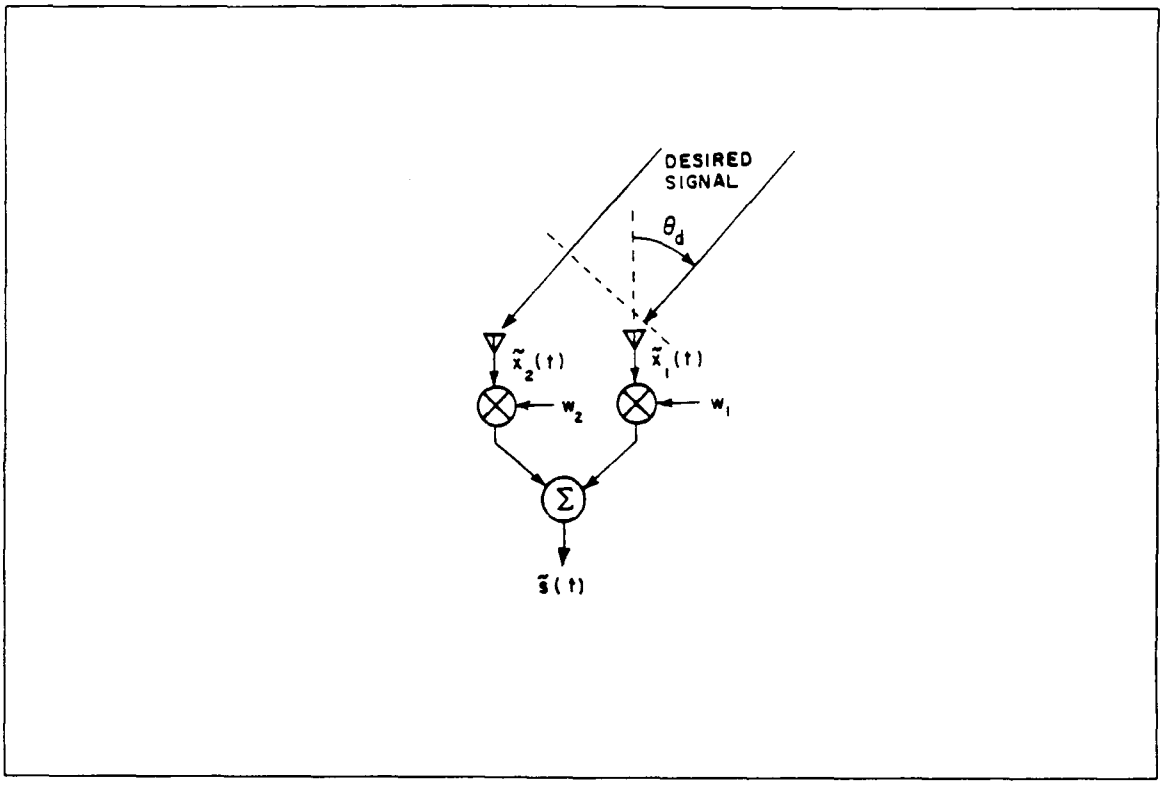

Figure 15. Two element array (4:24)

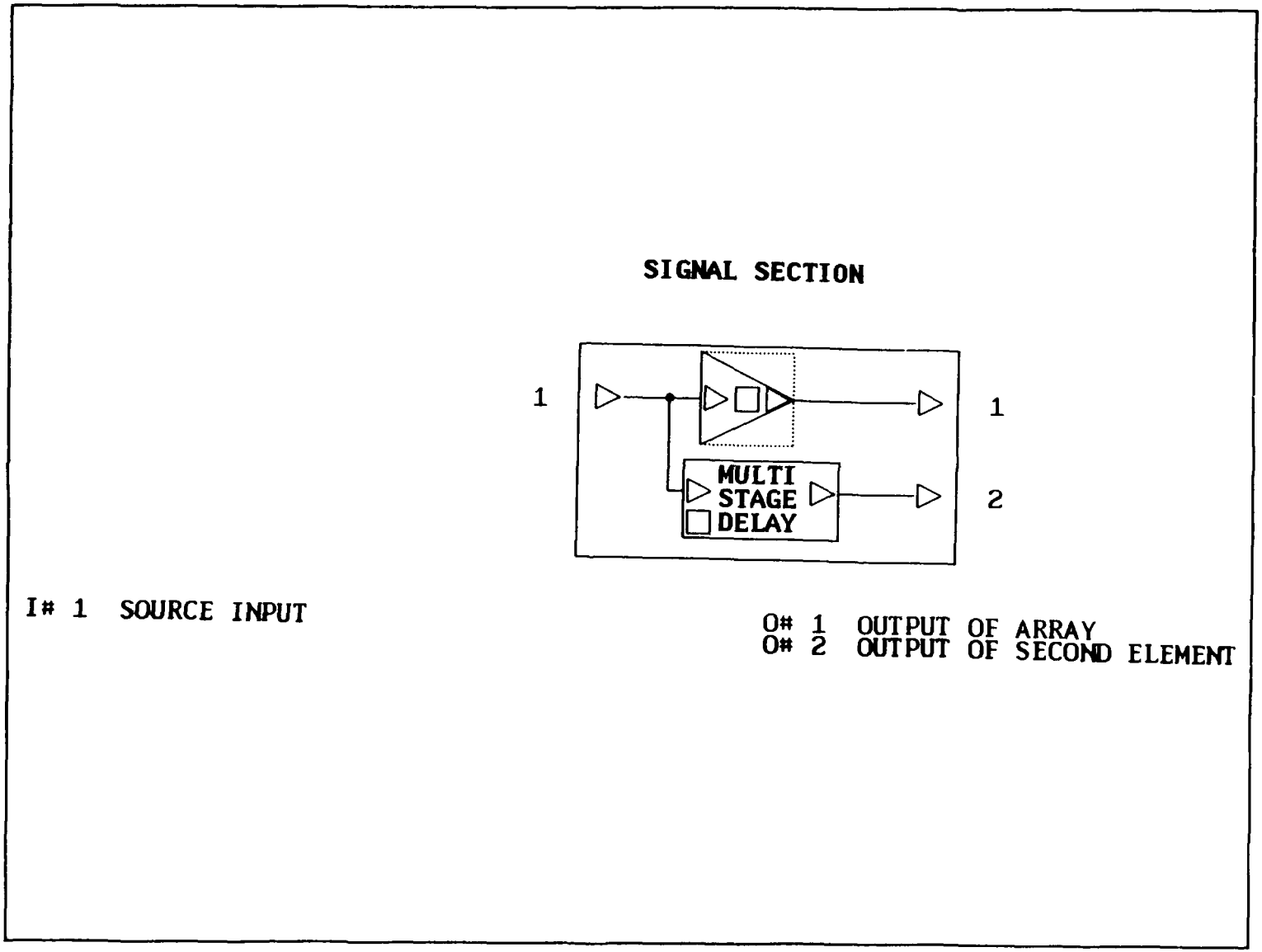

Figure 16. BOSS Two Element Array Signal Section

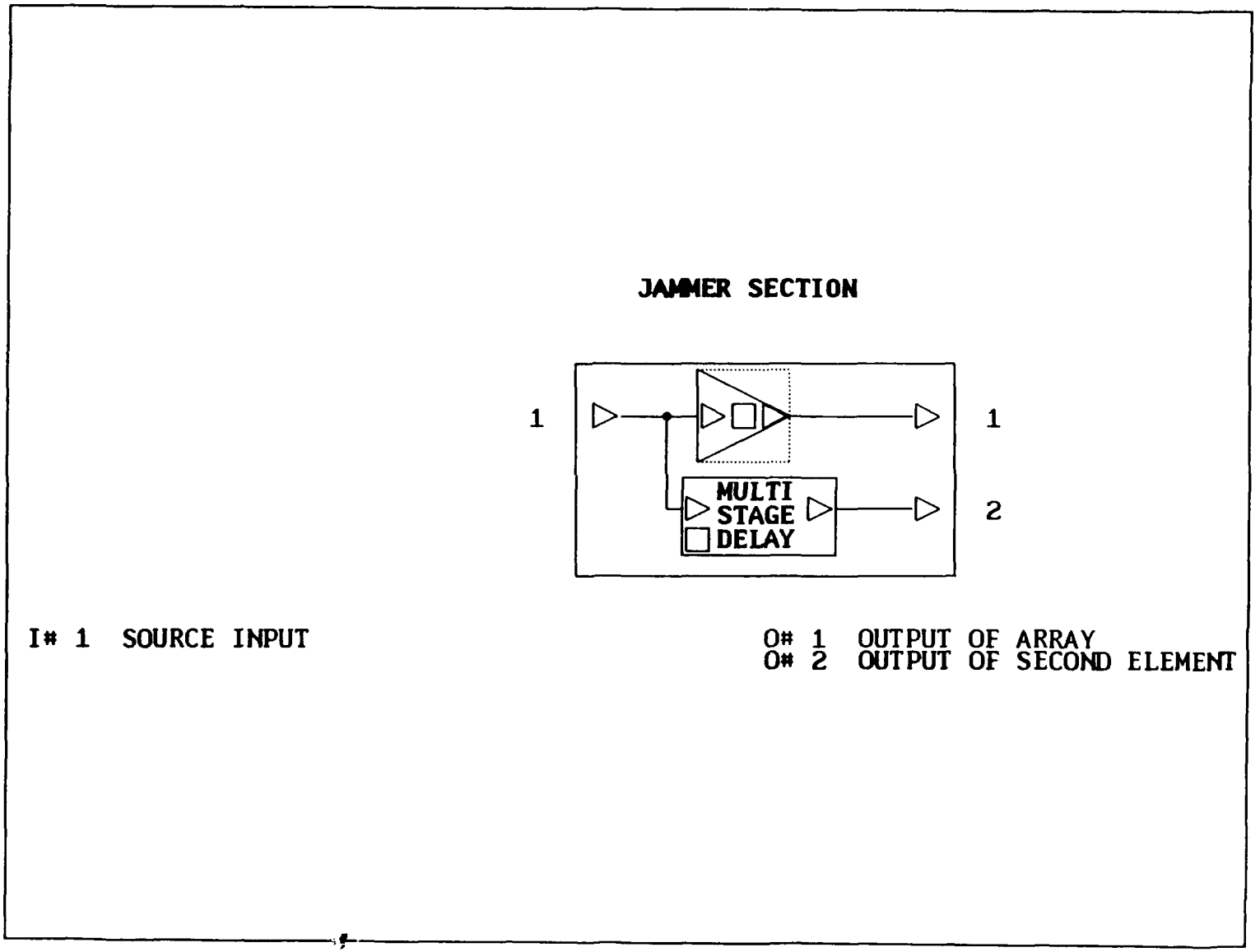

Figure 17. BOSS Two Element Array Jammer Section

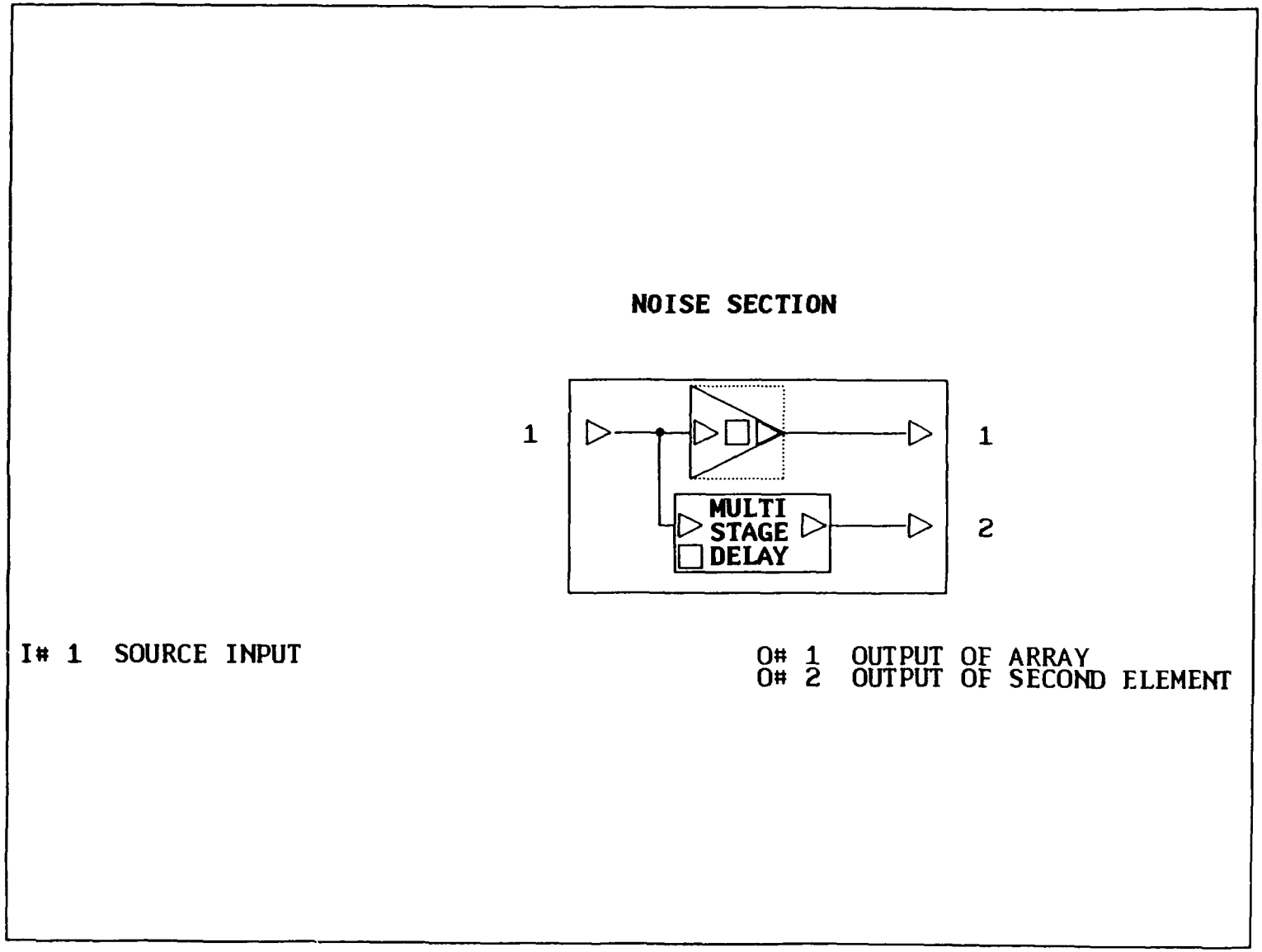

Figure 18. BOSS Two Element Array Noise Section

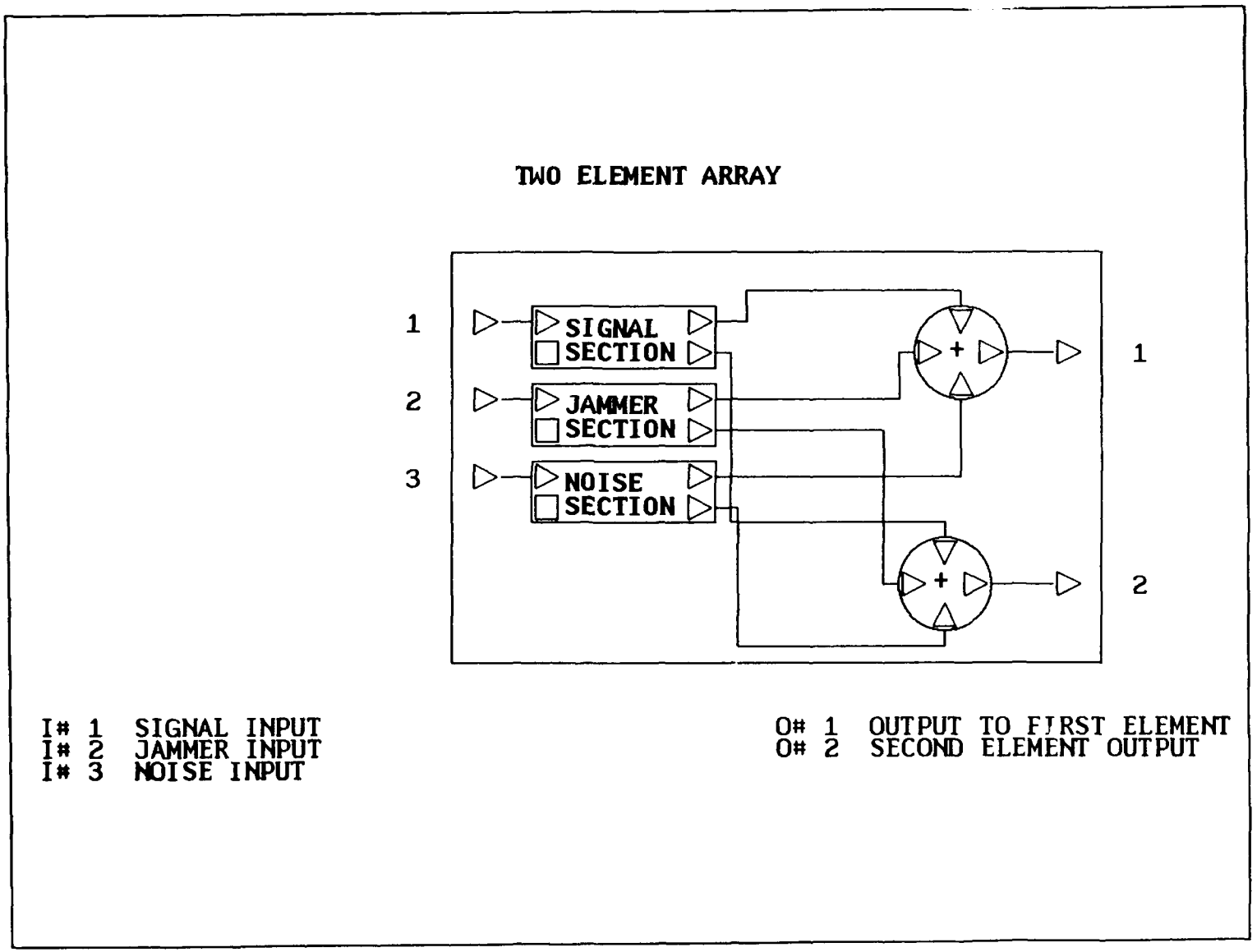

Figure 19. BOSS Two Element Array

Any number of array elements can be implemented in a similar fashion. The delay for each element is computed to provide the appropriate phase shift. The implementation for a four element array signal, jammer, and noise sections are shown in Figures (20), (21), and (22).

The arrays built in the previous section cannot handle complex signals. The idea is basically the same however. A phase shift must be provided to elements other than element 1. BOSS has a complex exponential module which produces an output of the form e^{jx} where x an input parameter. By multiplying a complex signal by the complex exponential, a phase shift is produced. To form the correct phase shift for each element, a constant value can be supplied for the value x. BOSS contains a *Constant Generator* module in its *Analog Sources* section. Using BOSS's complex exponential, constant generator, multiplier and gain modules, the signal, jammer and noise sections for both a two and four element arrays were constructed and are shown in Figures (23) through (28).

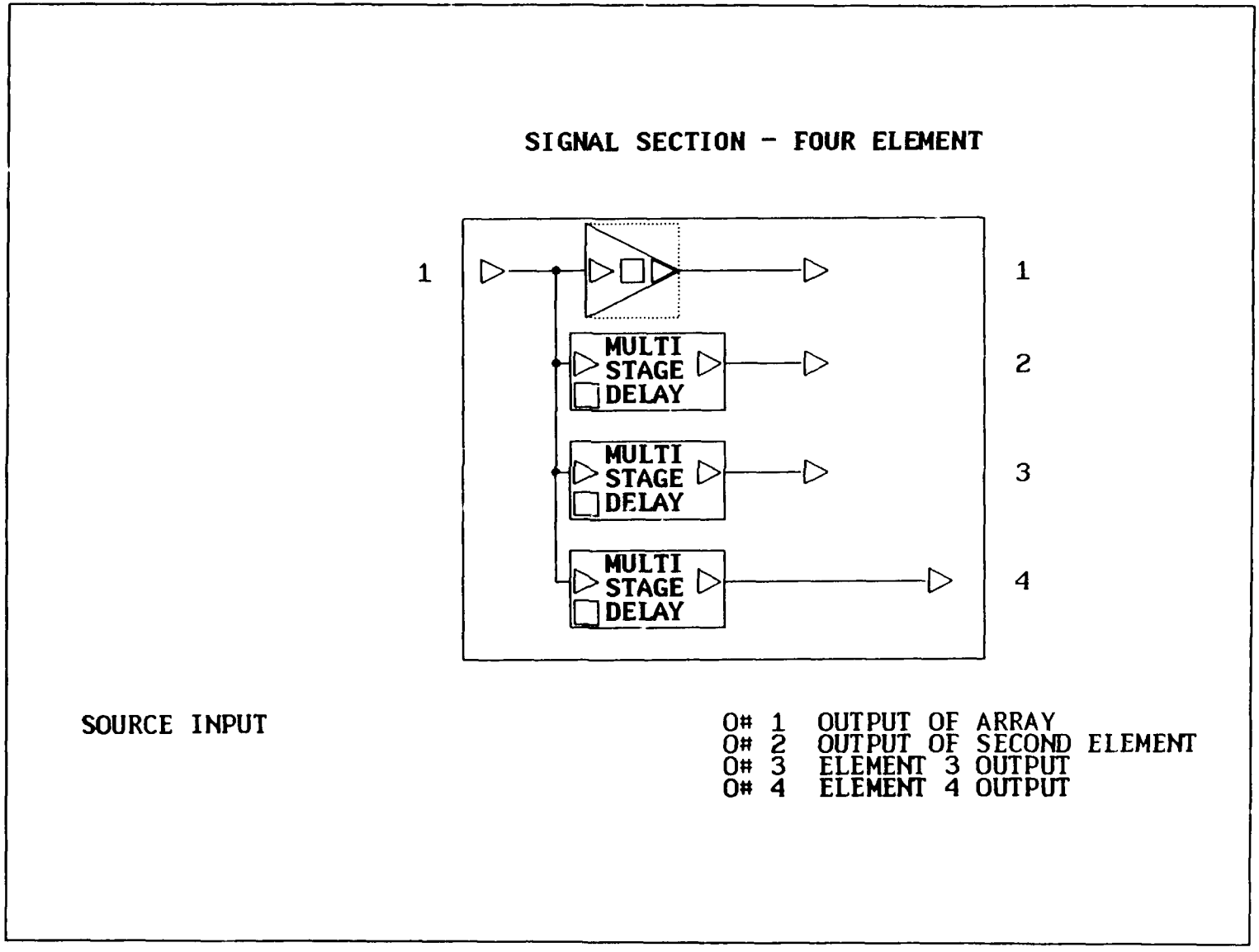

Figure 20. BOSS Four Element Array Signal Section

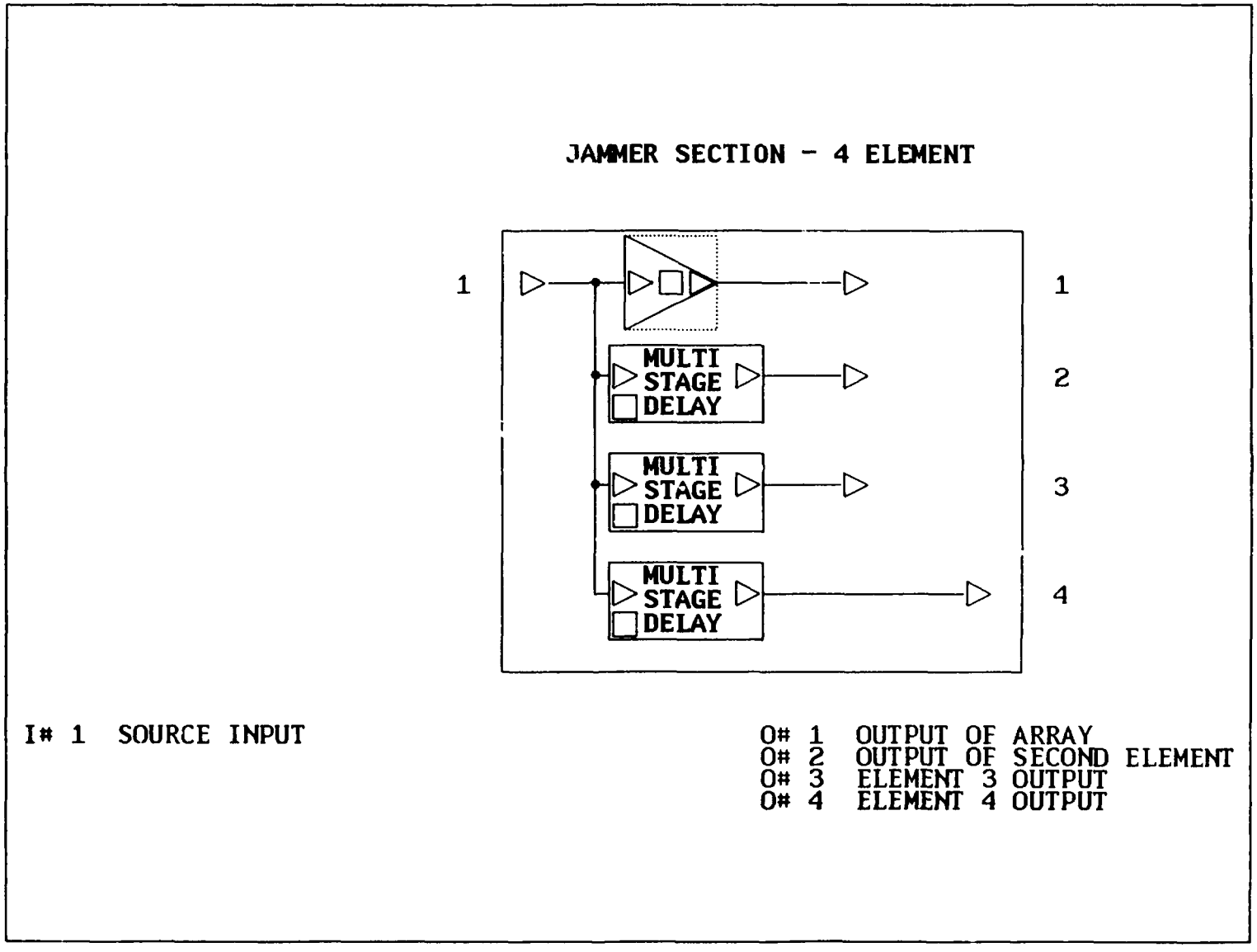

Figure 21. BOSS Four Element Array Jammer Section

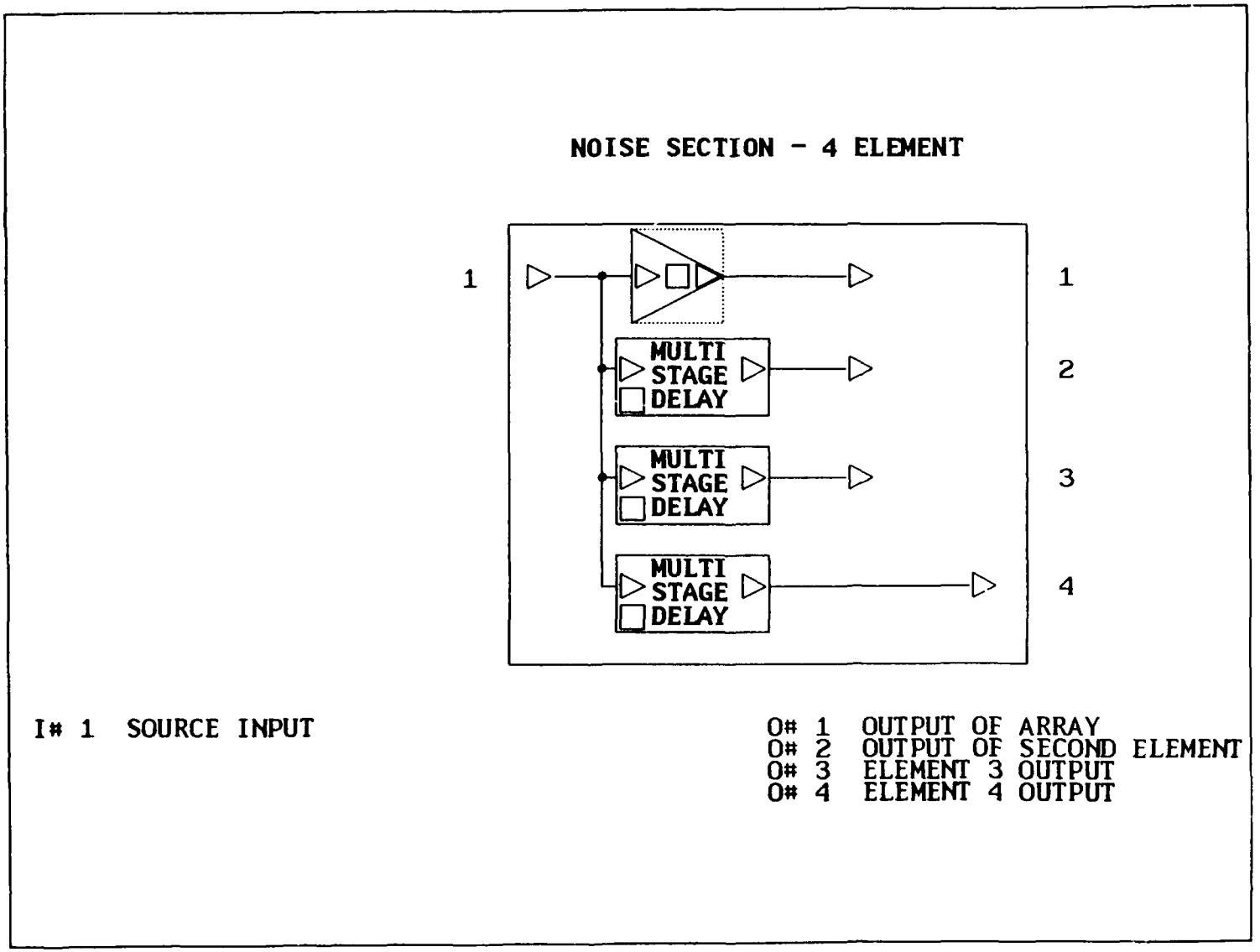

Figure 22. BOSS Four Element Array Noise Section

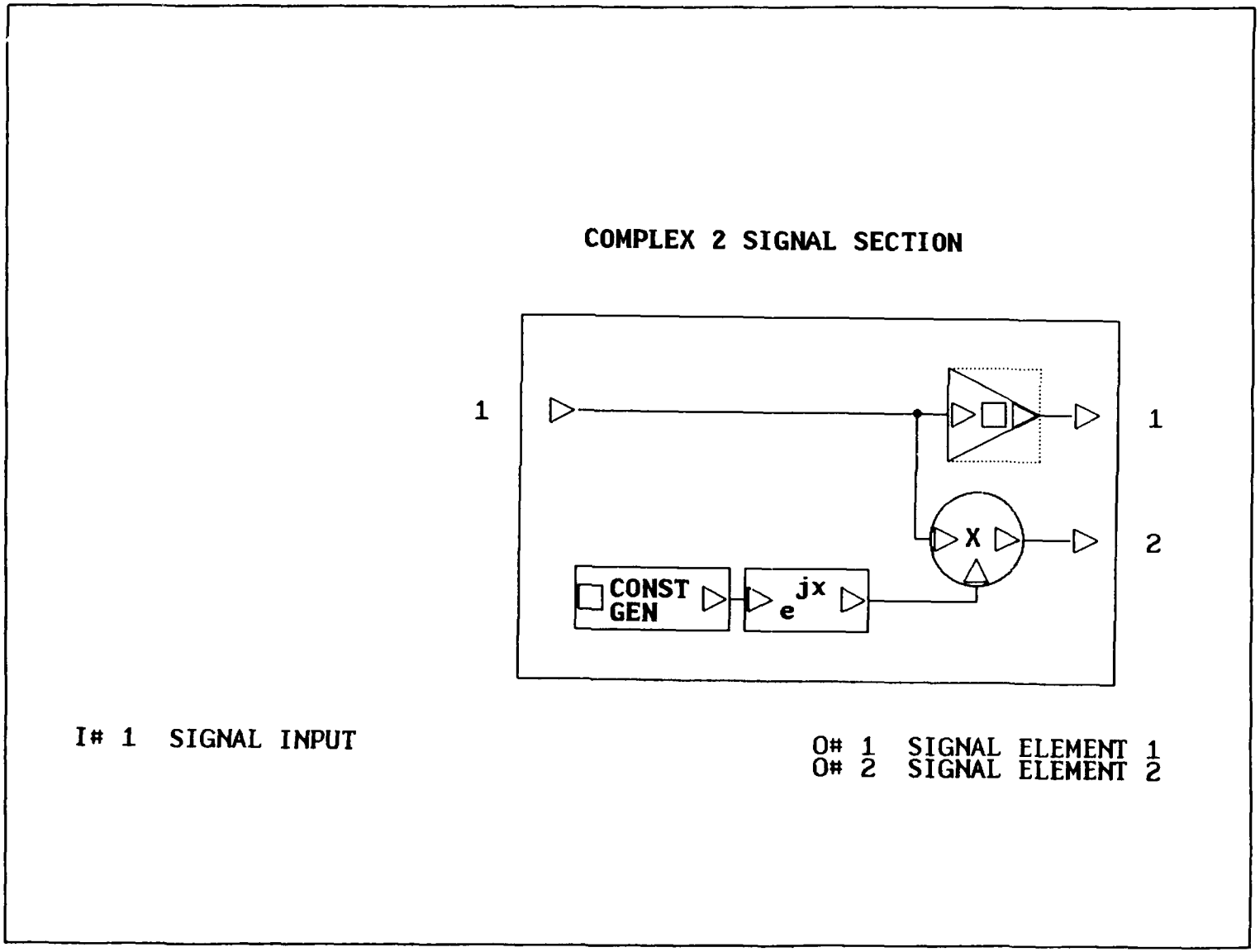

Figure 23. *BOSS Complex Two Element Array Signal Section*

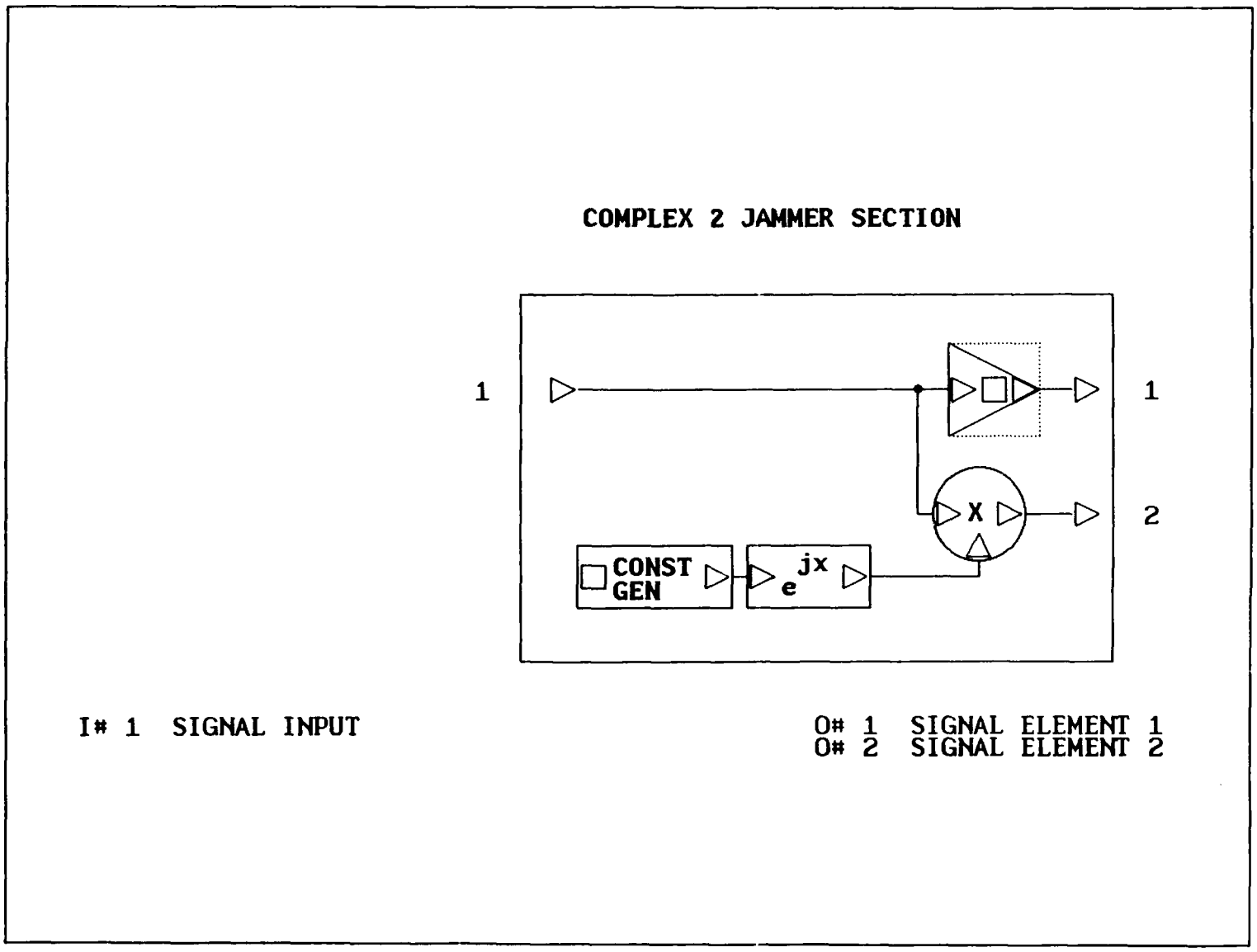

Figure 24. BOSS Complex Two Element Array Jammer Section

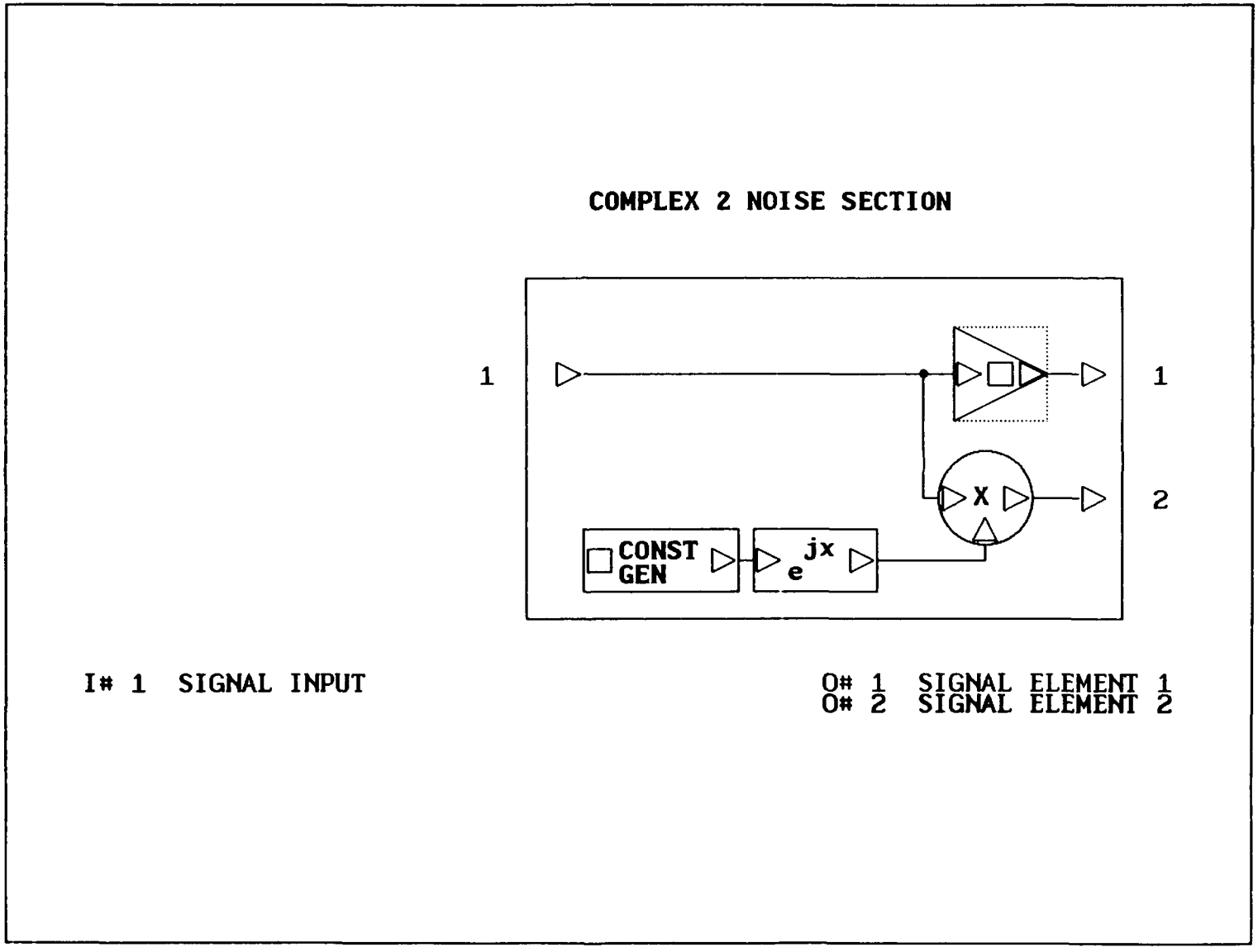

Figure 25. BOSS Complex Two Element Array Noise Section

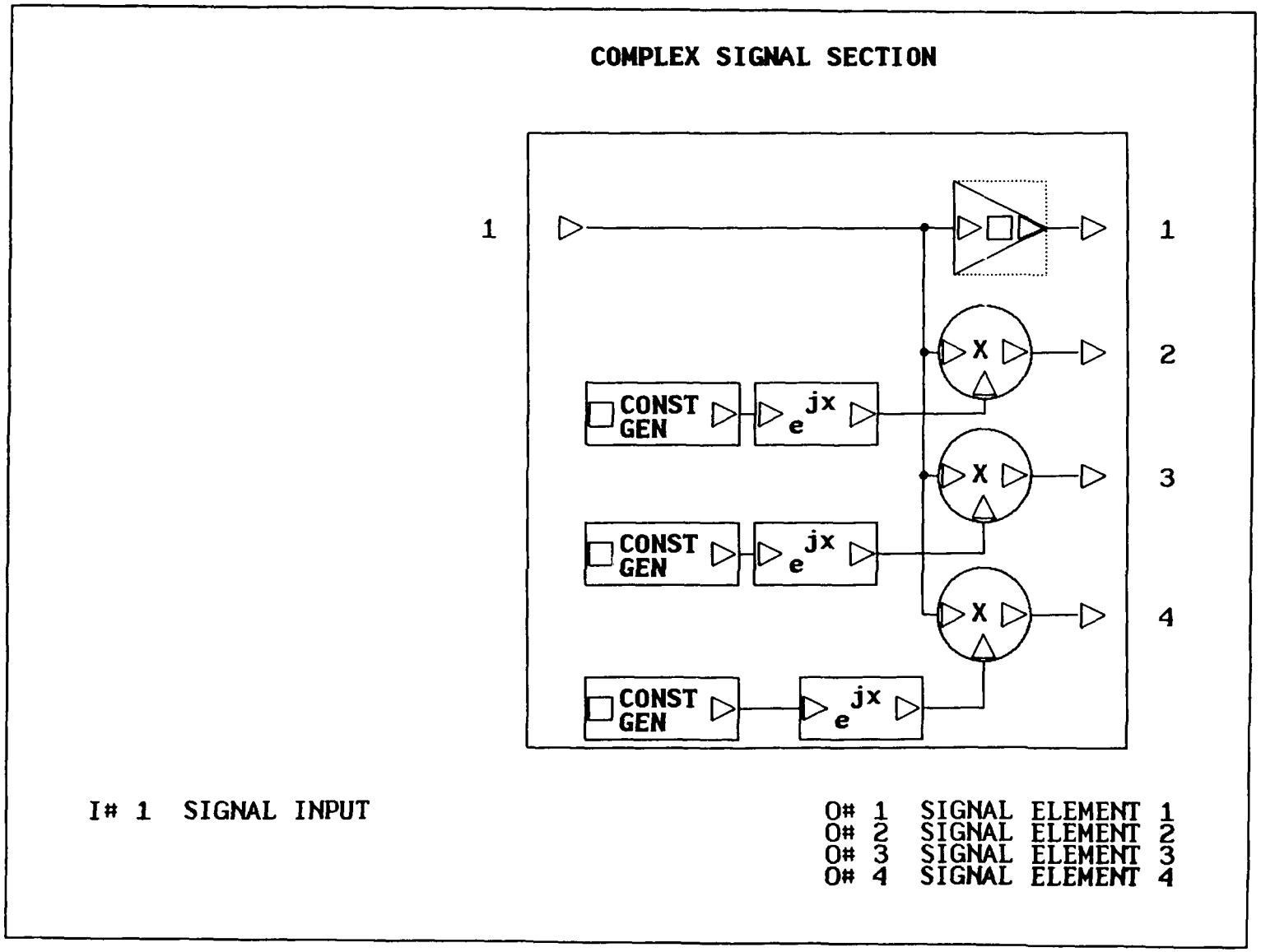

Figure 26. BOSS Complex Four Element Array Signal Section

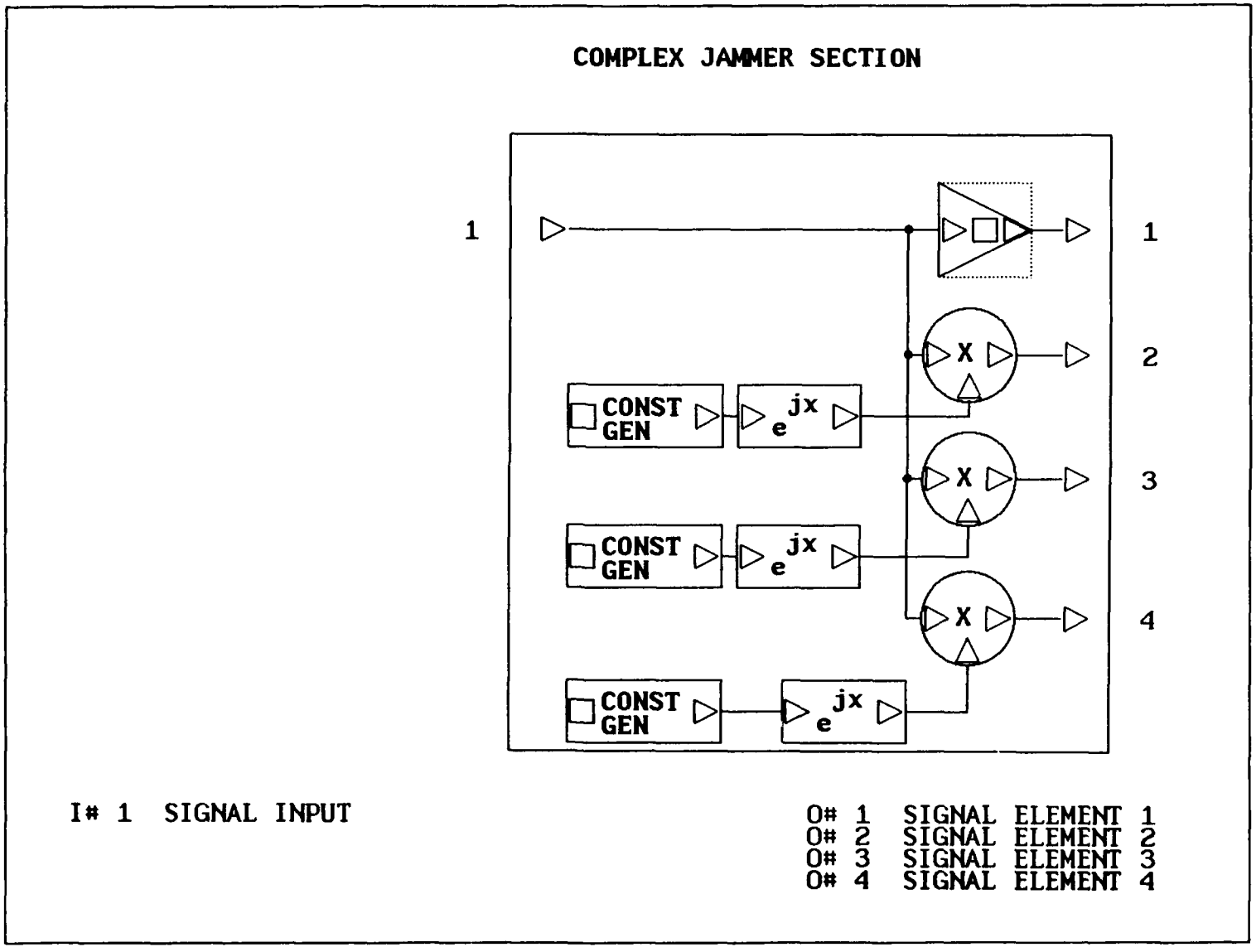

Figure 27. BOSS Complex Four Element Array Jammer Section

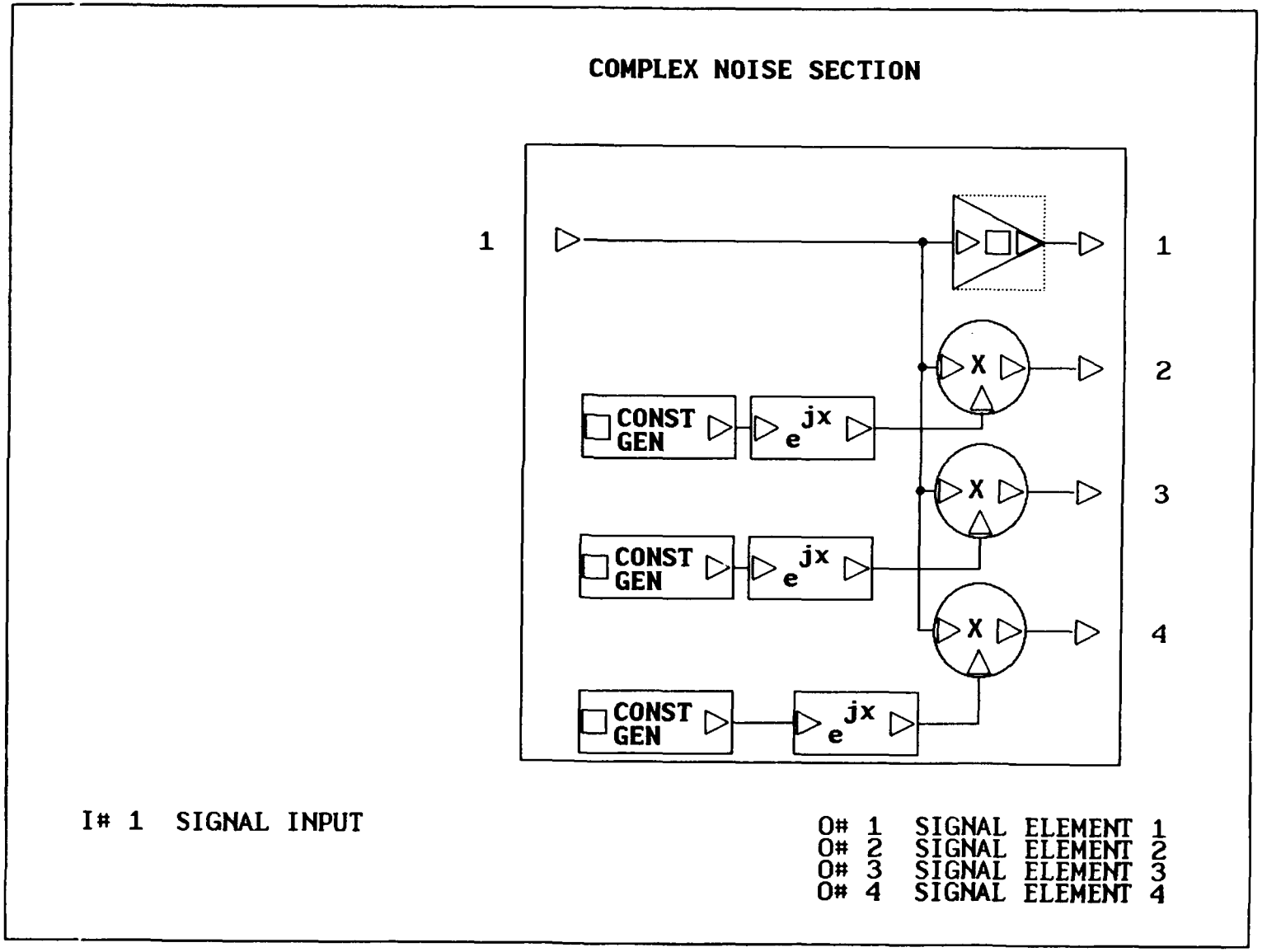

Figure 28. BOSS Complex Four Element Array Noise Section

3.6 Summary

With the algorithms and arrays implemented, the next step is to test each algorithm individually to see if weights correctly adapt. If this is successful, the algorithms will be put into systems with the arrays in an attempt to track desired signal while rejecting unwanted jammers. The results of these experiments are discussed in the next chapter.

IV. Simulation Results

4.1 Discrete LMS Algorithm Tests

The first step in testing the LMS algorithm implementation in BOSS, as with other algorithm implementations, was to build a simple test circuit in BOSS. A sinusoid or DC signal could be used as the input to the LMS loop. The same input signal could also be routed into an amplifier or attenuator and then used as the desired signal. After adaptation, the weight value should match the the amplifier's setting thereby modifying the input signal to match the desired signal. The BOSS implemetation of such a test circuit is shown in Figure (29). In this circuit, the

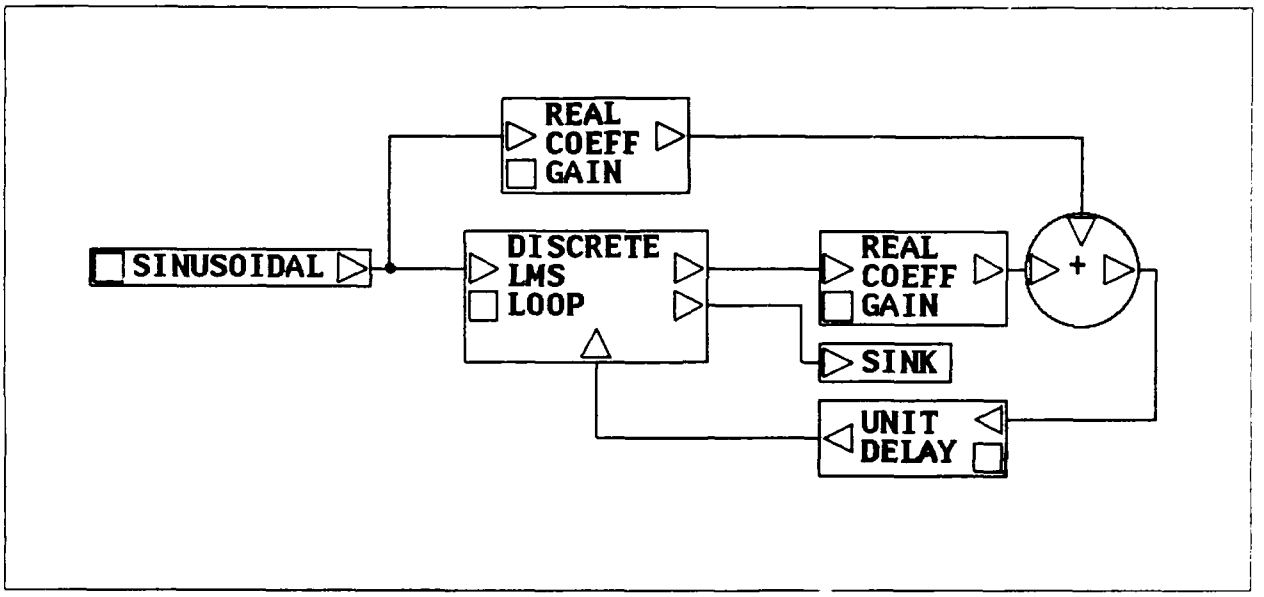

Figure 29. BOSS LMS Test Circuit

module labeled *REAL COEFF GAIN* at the top of the circuit is an amplifier that will modify the sinusoidal input to form the desired signal. The second gain module is set to a constant -1 to subtract the output of the loop from the desired signal to form the error out of the adder. The *UNIT DELAY* module is inserted in the feedback path to take into account that the error signal cannot be fedback instantaneously.

(BOSS will not allow a 0 time delay in a feedback path.) This module simply delays the feedback for one simulation iteration. Its output is initially set to 0. The module labeled *SINK* is used to provide a path for the monitored weight value. The LMS loop should adapt to match whatever the gain of the amplifier is during the simulation.

During simulations with the test circuit in Figure (29), the convergence factor of the LMS loop, the amplifier gain factor, and the sinusoid's frequency, phase, and amplitude were exported as simulation parameters. Exporting these parameters allowed them to be varied during different simulations in order to observe the loops performance without constantly changing the circuit itself.

Appendix A contains BOSS signal plots of all circuits tested. Results for the LMS test circuit in Figure (29) are in Figures (37), (38), (39), and (40). In this first test of the discrete LMS circuit, the simulation parameters are shown in Table (1). With μ set to 0.1, the weight adapted to 0.5 in 1.2 seconds when the input was a 5 Hertz (Hz) sinusoid. The final weight result was exactly as expected with the desired amplitude at 1/2 of the input signal. Since the output was a duplicate of the desired signal, the error converged to 0 within the 1.1 seconds. The entire experiment was repeated for 5 KHz, 5 MHz and 5 GHz sinusoids. The results in Appendix A show the same adaption behavior but scaled by time. This result indicated that the discrete LMS loop contains no frequency dependent components. Whenever this LMS loop was used in a circuit, there was no reason not to operate at lower frequencies since the results can be translated to higher frequencies.

The next step in testing involved varying the convergence factor μ. Decreasing the convergence factor should lead to a smoother convergence of the weight while actual convergence takes longer. In the first test, μ was reduced for 0.1 to 0.05. As Figures (53), (54), and (55) show, convergence time did increase to approximately 2.5 seconds. However, the convergence was smoother. The convergence factor was then increased to 0.5. The weight value did converge faster as shown in Figure (58). Convergence took approximately 0.4 seconds versus 1.1 seconds when μ was set to

0.1 and 2.5 seconds with μ set to 0.05. However, there was a 0.1 overshoot which represents 20 percent of the weights final value. Increasing the convergence factor does increase the speed of convergence but at the price of overshoot. This type of behavior is exactly what is to be expected for an LMS loop. For a similar circuit, Compton shows weight convergence in approximately 100 samples with μ set at 0.1 (4:71). Table (1) shows a sampling rate of 100 samples per second and μ set to 0.1. For convergence in 1.1 seconds shown in Figure (40), 110 samples were taken. The same page showed convergence in 35 samples when μ was increased to 0.5. Figure (58) shows convergence in 40 samples when sampling at 100 samples per second. In both cases, the simulation produced by BOSS agreed closely with Compton's results.

For stability, the value of μ should obey the following equation (9:50):

$$0 < \mu < 1/\lambda \tag{53}$$

where λ is the sum of the eigenvalues in the R matrix in Eq (11). For the case of the 5 Hz sinusoid with an amplitude of 1, μ must be less than 1. As more sources are added to the circuit, the power into the circuit increases and μ must be decreased.

For all the testing at various frequencies and values of the convergence factors, the BOSS implementation of the discrete LMS loop adapted to the produce the desired output. The implementation of the loop and testing through simulation were easier than if the experiments has been programmed in a high order computer language.

4.2 Analog LMS Algorithm Tests

The BOSS discrete LMS loop in Figure (29) was replaced with the BOSS analog LMS loop in Figure (11). The resulting circuit is shown in Figure (30).

The results for the BOSS analog test circuit were exactly the same as the BOSS discrete test circuit. Convergence times, outputs, and weight values performed in

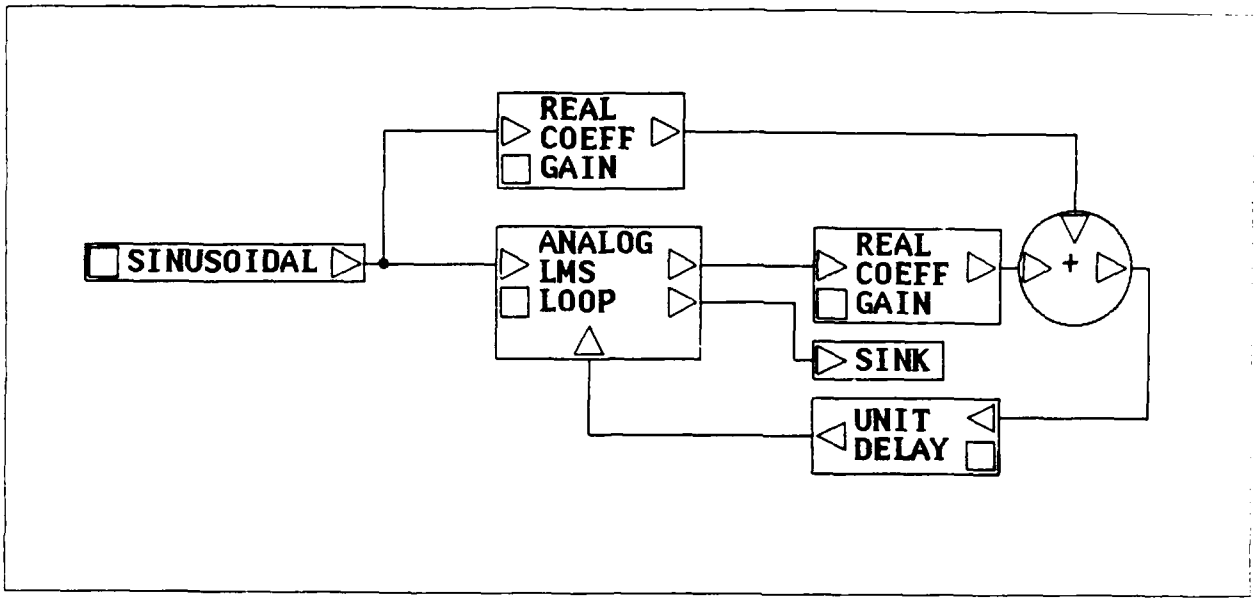

Figure 30. BOSS Analog LMS Test Circuit

exactly the same fashion as the discrete circuit. Therefore, it was arbitrarily decided to only use the discrete loop for further testing. Testing both the analog and discrete circuits was redundant.

4.3 Two Element Array Testing

With the BOSS LMS circuit tested, the next step was construction of a two element array system. The LMS loop in Figure (12) along with the two element array in Figure (19) were combined with other BOSS modules to form the system in Figure (31).

The top sinusoid input on the left side acts as the transmitter the system is trying to receive. The sinusoid near the upper right part of the system represents the desired signal. The sinusoid on the system's left acts as a tone jammer. The Gaussian random generator acts as a noise source. The two multi-stage delays are used in the system to provide the 90 degrees of phase shift needed to form the quadrature component of the loop inputs. The modules labeled *SINK* are used as a monitoring point for the weight values during simulation.

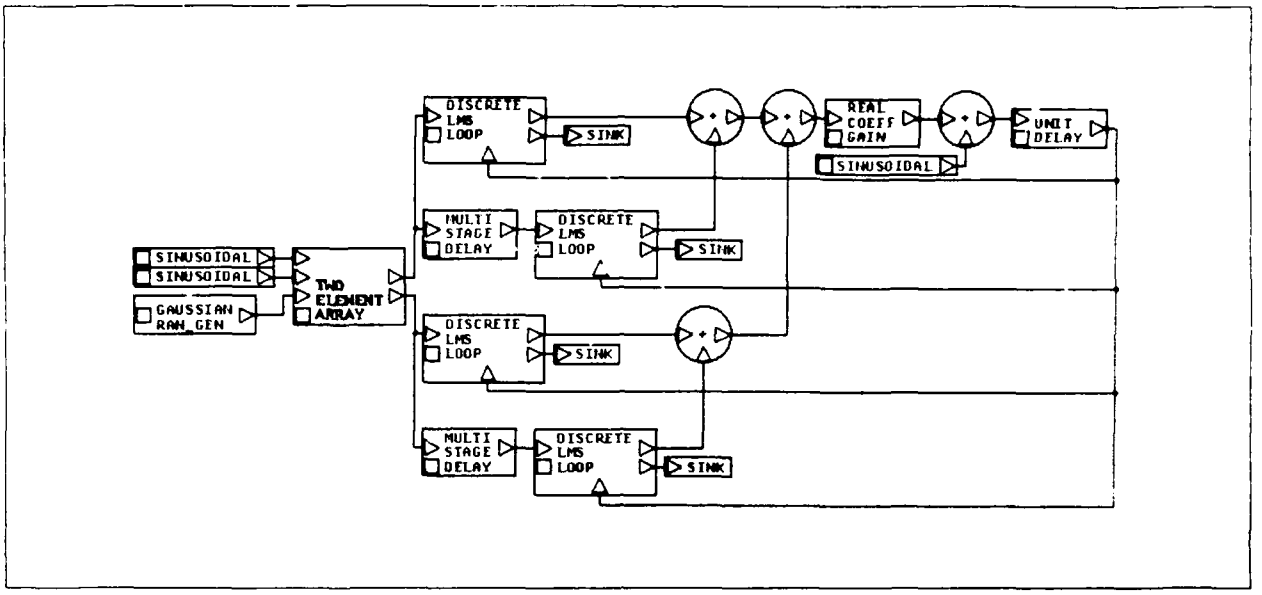

Figure 31. BOSS Two Element Array LMS System

The two element array LMS system was first tested with a 5 Hz sinusoid as the transmitter and the desired signal. The transmitter is broadside to the array. The parameters for this simulation are in Table (7) in Appendix A. The parameter labeled **DT** is the time between samples. In order to meet the Nyquist criterion of having the sampling frequency at least twice the highest frequency in the circuit, **DT** can be as great as 0.1 seconds for the 5 Hz input. However, with 0.1 seconds between samples, 2.5 samples must be used to provide the 90 degrees of phase shift for the quadrature weights. The number of samples to delay in the multi-stage delays must be an integer number. To achieve any integer degree phase shift, the sampling frequency must be higher (that is, **DT** must be smaller). By raising the sampling frequency to 1800 Hz (**DT** lowered to $5.55E-4$), each sample delay represents one degree of phase delay. To get 90 degrees of phase delay, the multi-stage delay was set to 90 samples. As can be seen in the output, error, and weight plots in Figures (59), (60), and (61), the system converged in 0.5 seconds. Since the input signal arrives at each array element with no relative phase shift, the inphase elements should track the signal with the quadrature elements going to 0. This is exactly what happened.

The gain in the direction of the source was 1 with each element providing 1/2 of the output. The convergence time of 0.5 seconds, (or 900 samples), again agrees within 5 percent of the convergence time of a two element system by Compton which used the LMS algorithm (4:71).

The source was then moved to 30 degrees relative to broadside of the arrary. Again a 5 Hz sinusoid was used as the input and desired signal in an attempt to adjust the phase of the array elements to "point" toward the source. The parameters for the simulation are in Table (8) in Appendix A. According to Eq (52), the input signal should be advanced 90 degrees at the second element. The numerical gain of the array behaves as a function of ϕ in the following equation:

$$\begin{aligned} Gain &= [w_{I1} + w_{I2}cos(\pi sin\phi) + w_{Q2}sin(\pi sin\phi)]^2 \\ &\quad + [w_{Q1} + w_{Q2}cos(\pi sin\phi) - w_{I2}sin(\pi sin\phi)]^2 \end{aligned} \tag{55}$$

where w_{In} and w_{Qn} are the inphase and quadrature weight values from Figure (62) in Appendix A. Putting these weight values into Eq (55) produces a maximum gain of 1 at 30 degrees. The LMS algorithm adjusted the weights until the output signal matched the desired signal producing a gain factor of 1 at 30 degrees. This simulation was duplicated many times while moving the source to different angles with respect to the array and varying the input signals amplitude. In every case, the LMS loops adapted to provide the correct gain in the appropriate direction. Moving the source affected the convergence time depending on how much phase adjustment of the weights was necessary. If the initial conditions of the weights (the inphase and quadrature components were always preset to 1), matched the amount of phase shift required at the end of adaptation, convergence time was almost zero. However, if the weights have to swing through 90 degrees of phase shift convergence time increased.

The final test of the two element array system involved including a tone jammer. The jammer was simply a sinusoid at the same 5 Hz frequency as the desired

source. A phase shift was provided for the jammer in the same manner as it was for the source in the previous experiment - by providing a delay in the signal at the second array element to simulate the phase shift between elements. The source was placed at 0 degrees and the jammer at 30 degrees relative to the array. The parameters for the simulation are contained in Table (9). During the first several simulations with this configuration, the array was steered toward the jammer. Since the jammer is a 5 Hz sinusoidal signal just like the desired signal, there is correlation between the jammer and the desired signal. As mentioned in Chapter 2, the interference source should not be correlated with the desired signal if the array is to reject the jammer. Since there is correlation between the jammer and the desired signal, it was decided to delay the jammer signal several seconds and have the weights "trained" in the direction of the source. This was accomplished by putting a multi-stage delay at the jammer output and having the number of samples as a simulation parameter. Since there was an overall increase in power entering the array, the convergence factor had to be reduced from 0.1 to 0.005 to prevent divergence of the error and output signals. The plots of the array output and error show a slower convergence toward a zero error. Using Eq (55) and the weight values from the plots with the jammer, the two element array only achieved approximately a 3 dB suppresion of the jammer signal relative to the desired input. The 3 dB figure was computed by first using the weight values when the source was placed at 0 degrees with no jammer and computing the gain at 0 and 15 degrees using Eq (55). Multiplying these numbers by the amplitude of the sinusoids at 0 and 15 degrees provides an unadapted signal to jammer ratio. The process was repeated with the weight values after adaptation to provide the 3 dB figure. This number agrees exactly with the suppression obtained with a two element array simulated by Compton (4:37).

4.4 *Four Element Array Testing*

With the two element array testing completed, a four element system was implemented with modules for the four element array in Chapter 2. The approach was similar to that for a two element array system. Eight LMS loops were connected to the four element array, the loop outputs summed to form the total array output, the total array output subtracted from the desired signal to form the error, and the error fed back to each loop. The resulting circuit is in Figure (32). The ability to

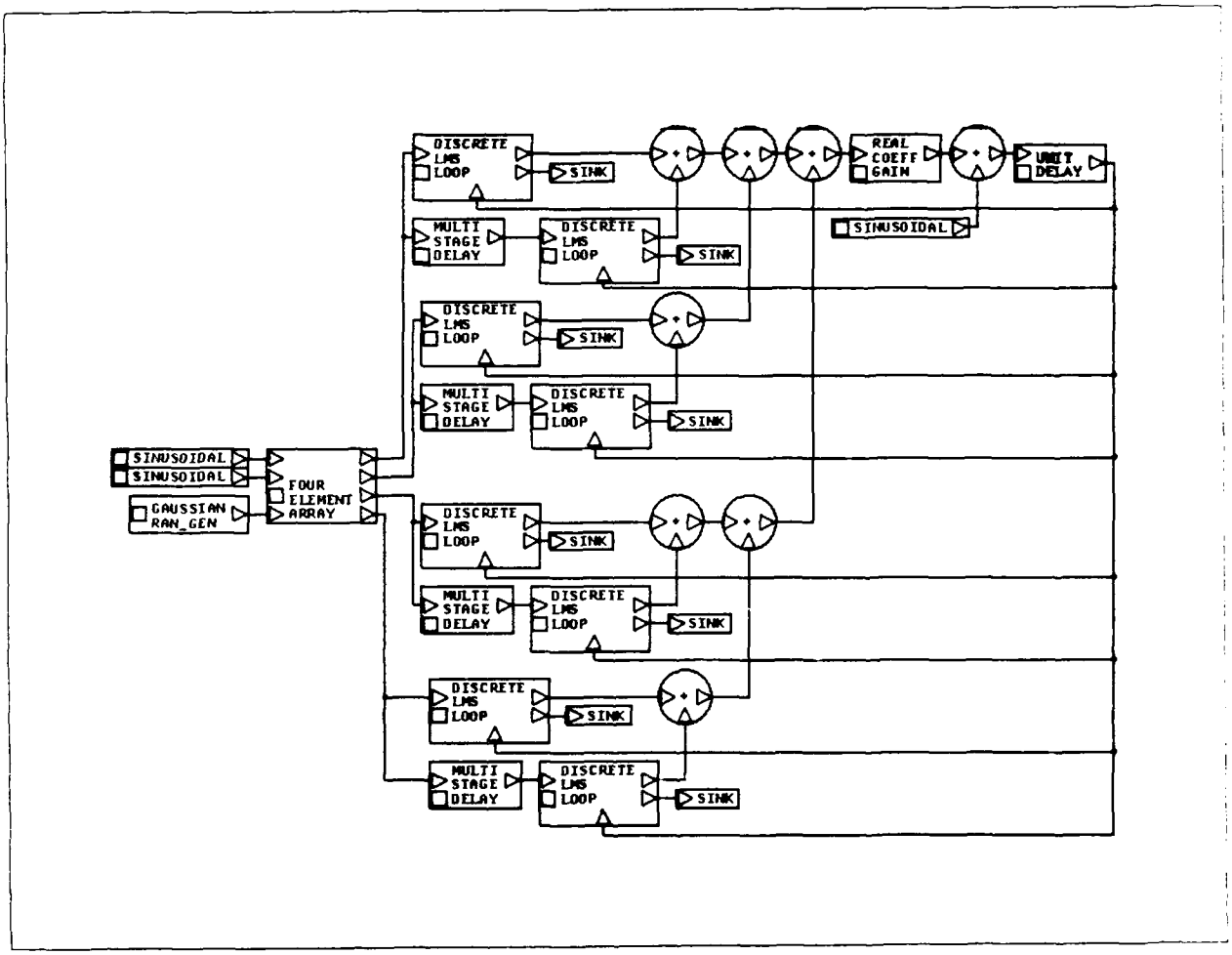

Figure 32. BOSS Four Element Array LMS System

delay the incoming signal, jammer, and noise any number of samples really allows for arbitrary phase shifts between array elements. This capability permits any geometry

or spacing to be simulated between array elements. A square geometry was chosen with the sides of the square half a wavelength long. If the upper left corner of the square contains the reference element. upper right corner element has a $\pi sin\theta_d$ phase shift compared to reference element, the lower left element experiences a $-\pi cos\theta_d$ phase shift, and the lower right element has a $\pi sin\theta_d - \pi cos\theta_d$ phase shift compared to the reference element.

With the source at 0 degrees, Figure (68) shows the weights converging in approximately 0.5 seconds. In the figure, I1, I2, I3, I4 are the inphase weights while Q1, Q2, Q3, Q4 are the quadrature weights. Expanding Eq (55) to include the additional weights, the values for the weights in Figure (68) produce a gain of 1 at 0 degrees. With the source at 30 degrees, the weights converged to the values in Figure (69). These values produce a gain of 1 at 30 degrees. Adding a jammer to the system produced the most interesting results. A multi-stage delay was again needed to allow the weights to first converge on the desired source before adapting in an attempt to reject the jammer. The weights in this simulation took almost seven seconds to converge to their final values shown in Figure (72). These weight values produced 19.5 dB improvement in signal to noise ratio. In a similar experiment, Widrow used a twelve element array with multiple jammer sources. This arrangement produced -26 to -38 dB sensitivity in the jammer direction relative to the source direction (10:2154). In this thesis, the jammer power was only 19 dB more than the source. The array sensitivity in the jammer direction compared to the signal direction was exactly the 19 db figure. Even though Widrow achieved more suppression, the interference sources had more power.

4.5 Complex LMS Algorithm Testing

A complex LMS test circuit was implemented using the complex LMS loop module. BOSS provides a complex tone generator module in its *ANALOG SOURCES* section and this tone generator was used as the input to the LMS loop. The ampli-

tude of the desired signal was produced by passing the input through a gain module where the gain was a simulation time parameter. The desired signal phase was formed by multiplying the input signal by a complex exponential. The complex exponential phase was constant provided by a constant generator which had a value set as a parameter at simulation time. Multiplying the input by the weight value, forming the error, and providing feedback were accomplished in a fashion similar to previous test circuits. The complex LMS test circuit is shown in Figure (33).

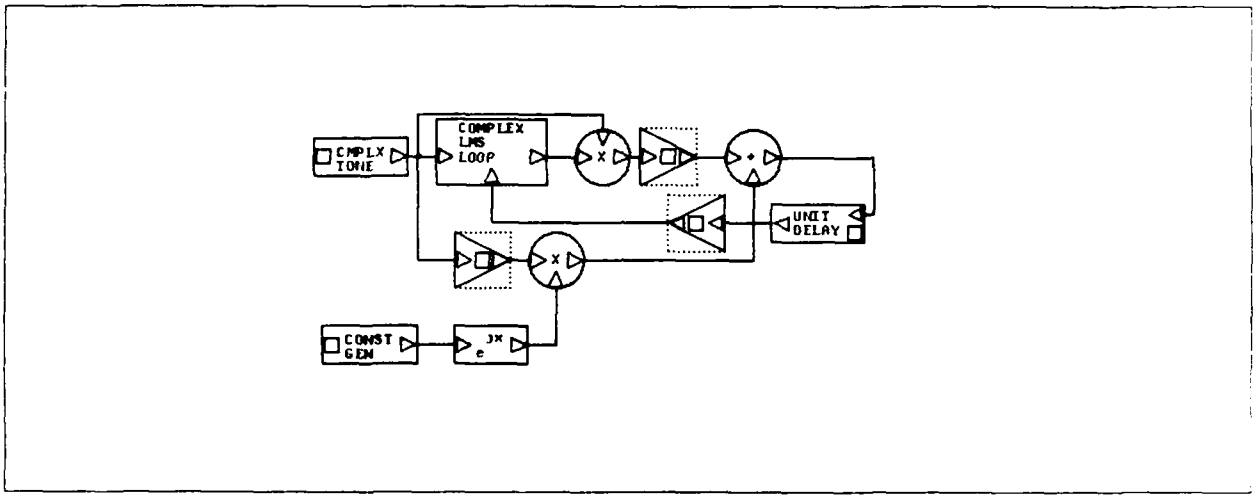

Figure 33. BOSS Complex LMS Test Circuit

Figures (76) and (77) show the real and imaginary components of the complex weight adapting to 0.5 and 0.0 when the input signal was multiplied by 0.5 with no phase shift to form the desired signal. When 30 degrees of phase were added to the input, the real component of the weight adapted to 0.433 while the imaginary component adapted to 0.25. This produced a 30 degree phase shift of the input and a reduction of the inputs magnitude of 0.5. The complex weight adaption is shown in Figures (78) and (79). The complex test circuit showed that the complex LMS loop implemented in Figure(13) adapted to simple inputs where the amplitude and phase of the desired signal were not the same as the input.

4.6 Complex Four Element Array Testing

A four element array system capable of handling complex signal was implemented using the array section described in Chapter 3. The array inputs were two complex tones, one acting as the transmitter and the other as a tone jammer, and complex white noise. All three sources are available in BOSS's *ANALOG SOURCES* section. After both complex tones, a gain module was added to provide attenuation of the input and jammer signals. Each of the array's four outputs were fed into a complex LMS loop and then multiplied by the loop's output which is a complex weight. The weighted array outputs were all summed to form the adapted array output. An error signal was again formed by subtracting a desired complex tone from the adapted array output. The error was then fed back to each complex LMS loop. The resulting circuit is shown in Figure (34).

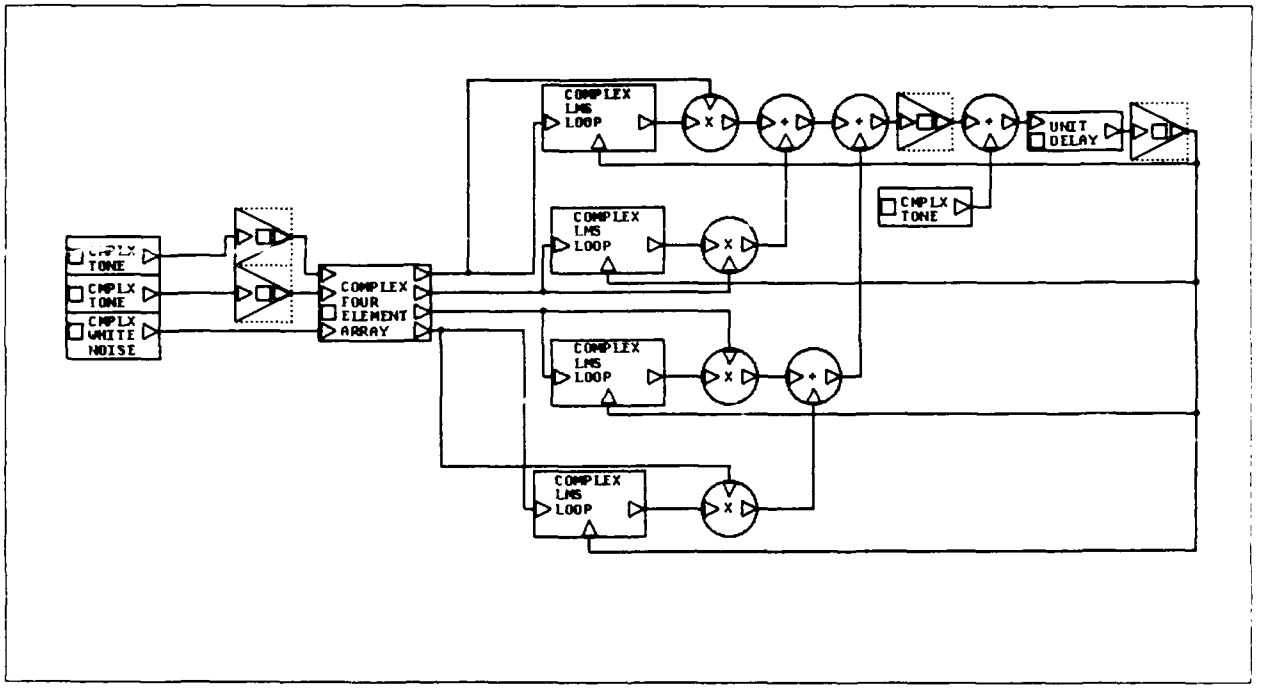

Figure 34. BOSS Complex LMS Test Circuit

Three simulations were performed. Just as in the past, the source as positioned as 0 degrees and then 30 degrees for the first two simulations. The final test was

with the source at 0 degrees and a jammer at 30 degrees. Each simulation results along with a table for the parameters of each similation are in Appendix A. During each simulation, probes were placed at the input to the array, at the weighted array output, at the error signal, and at each complex weight. Plots are shown for the magnitude of the output and errors along with the real and complex components of the weights. As can be seen in Figures (80), (81), and (82), the complex weights in this system adapt until the output matches the desired signal. The desired signal was a complex tone at 5 Hz with no phase shift for this simulation. Using an expanded form of Eq (55), the weights put gain of 1 in the source's direction. Placing the source at 30 degrees caused the weights to adapt to place a gain of 1 at 30 degrees. The plots for the error, output, and weights are shown in Figures (83), (84), and (85). The weights converged in approximately 0.3 seconds for both simulations. Even though this is a four element system, the convergence time is within ten percent of Compton's two element system (4:71).

A tone jammer operating at 5 Hz was added for the last simulation of this system. The jammer was placed at 30 degrees clockwise referenced to the upper left array element. The weights again adapted to produce an output which is a reproduction of the desired signal. However, the weights took almost 1 second to adapt because the convergence factor had to be reduced when the power increased at the array input. The signal to noise ratio was improved by 19 dB just as it had for the four element real system.

4.7 Applebaum Algorithm Testing

The BOSS Applebaum loop implementation shown in Figure (14) was tested using a sidelobe canceller circuit suggested by Gabriel (5:242). The test arrangement is shown in Figure (35).

A complex tone source is added to a complex white noise source to form the block labeled *Source + Noise*. This block's output acts as a tone jammer with noise.

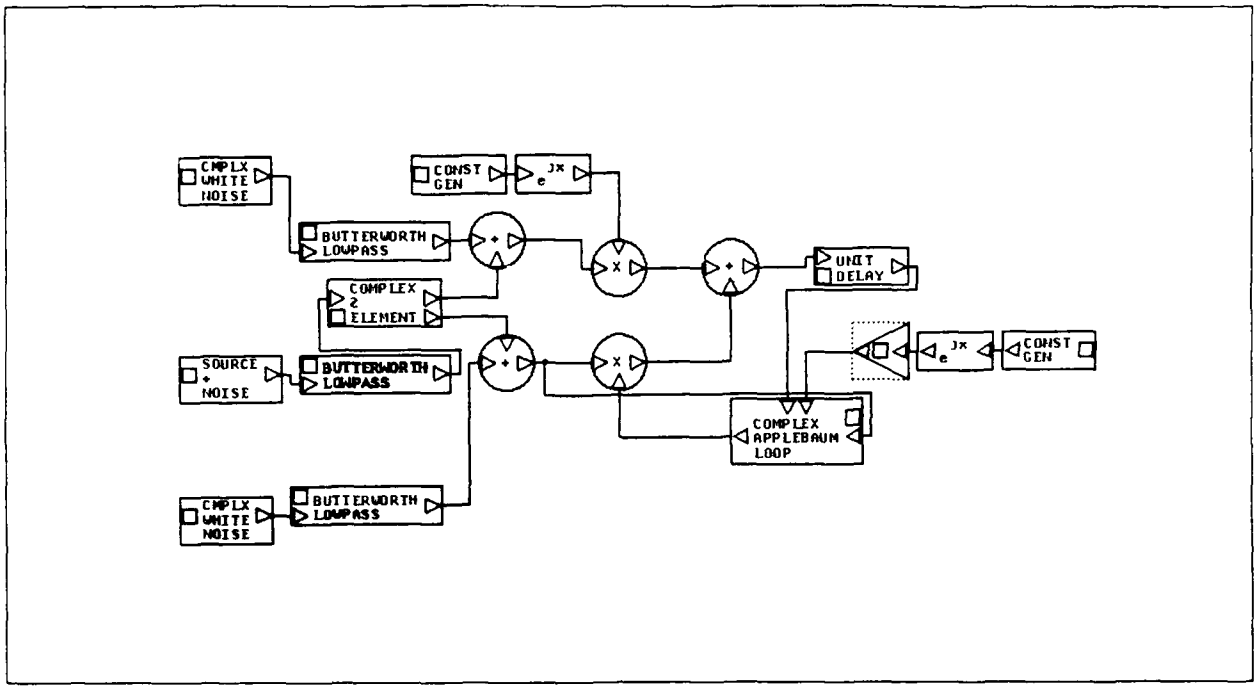

Figure 35. BOSS Applebaum Test Circuit

Each channel also has uncorrelated complex white noise added to simulate channel noise. The output of the top channel signal is added to the bottom channel output to form the sidelobe canceller output. The bottom channel input is multiplied by the complex weight formed by the block labeled *Complex Applebaum Loop*. The array output and the t_k component are fed into the loop along with the bottom channel input. All input signals are lowpass filtered to simulate the passband of a receiver. The loop lowpass cutoff frequency was computed by the following formula (5:245)

$$\tau_o = \frac{10}{\pi B_c}(1 + \mu + \mu P_i) \tag{56}$$

where τ_o is the time constant of the loop lowpass filter, B_c is the cutoff frequency of the input lowpass filters, μ is the convergence factor, and P_i is the ratio of interference power to receiver channel noise power.

For the first test of this circuit, a tone jammer 20 dB above the channel noise was placed at 10 degrees relative to broadside of the array. The steering vector, t_k,

was set to a desired look direction of zero degrees. The input cutoff frequency was set to 5 MHz and the loop lowpass frequency set to 15.4 KHz. Using these numbers, the complex weight should adapt to an optimal value of $0.96e^{j148^{\circ}}$ (5:246). Figures (89) and (90) in Appendix A show the complex weight converging to the ideal value before diverging. The input magnitude spectrum in Figure (91) of Appendix A shows an input jammer at 50 Hz with 20 dB of power. The output spectrum shown in Figure (92) shows a 16 dB decrease in power at 50 Hz. Putting the weight value in Eq (55) produces the array plot in Figure (93). This plot shows a 16 dB null at the 10 degree point. However, the weight did not remain at this point and continued to diverge. Different values of the loop lowpass frequency along with varying values of μ were used in an attempt to keep the weight from diverging. There is possibly a problem with the BOSS implementation of the Butterworth lowpass filter that accounts for the weight divergence.

In an attempt to correct the weight divergence, a discrete implementation of the Applebaum loop suggested by Griffiths was implemented. The weights in this implementation are updated according to the following equation (6:1696)

$$w_i(n+1) = w_i(n) + \gamma\mu t_i^* - \gamma x_i^*(n)s(n) \tag{57}$$

where $w_i(n+1)$ and $w_i(n)$ are the next iteration and current iteration weight values, μ is the usual convergence factor, t_i^* is the complex conjugate of the steering vector for the ith array element, $x_i^*(n)$ is the complex conjugate of the ith input, $s(n)$ is the summed array output and γ is a constant set according to

$$0 < \gamma < \frac{2}{P_{im}} \tag{58}$$

This loop was implemented in BOSS and substituted for the *Complex Applebaum Loop* in Figure (35). A simulation was again performed with a jammer located at 10 degrees relative to broadside to the array. The parameters for the simulation are

in Table (18). The weight converged to a value of 1.0 with a phase of 148 degrees as shown in Figures (94) and (95). The magnitude spectrum for the input shown in Figure (96) has a 20 dB jammer located at 50 Hz. The output spectrum in Figure (97) shows the total elimination of this jammer. Again using Eq (55), the array gain plot is shown in Figure (98). There is a 21 dB null at 10 degrees which is the location of the jammer. The theoretical value from Gabriel (5:246) produces an infinitely deep null at 10 degrees. But as Compton mentions, the nulls are only as deep as the jammer power (4:38). The simulation results for the discrete Applebaum loop match exactly the theoretical predications. With a working loop tested, the next step was using the loop in the four element array.

4.8 Complex Four Element Array with Applebaum Loops

The Applebaum loop was used with the four element array similar to Figure (34). The LMS loops were replaced with Applebaum loops. The steering vector components for each loop were formed with complex exponentials being multiplied by constant gain factors. The array's output is fed back to each loop instead of an error signal. The resulting circuit is shown in Figure (36).

The first test of this system placed a jammer at 15 degrees with the array elements arranged as a linear array rather than the square geometry used with the LMS loops. The steering vector was set for 0 degrees. The jammer power was 20 dB. Table (19) in Appendix A shows the parameters used for this simulation. The input and output spectrums in Figures (99) and (100) again show a 20 dB jammer at 50 Hz before adaptation and the elimination of the jammer after adaptation. Using the weight values from this simulation in an expanded from of Eq (55), shows a 20 dB null at 15 degrees. This matches exactly the theoretical prediction.

The second test of this system involved placing a second jammer at -30 degrees relative to broadside of the linear array. A 0 degree steering vector was again used with the parameters for the simulation shown in Table (20). The input spectrum

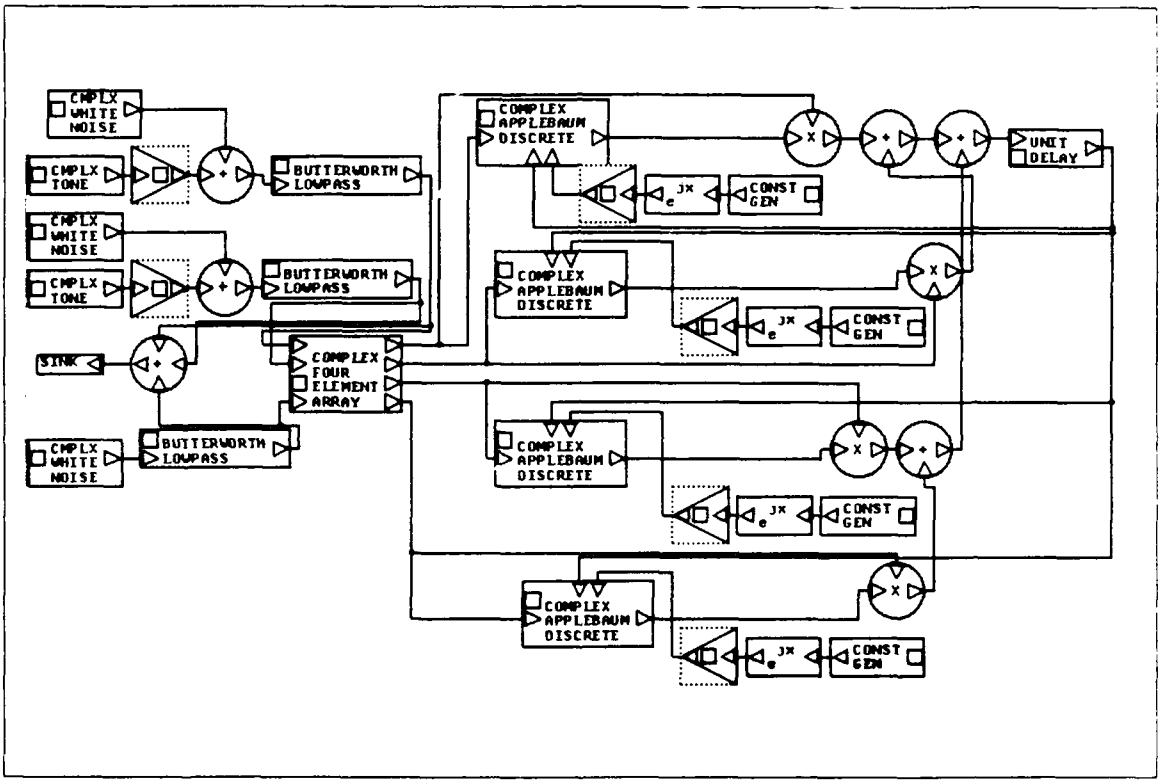

Figure 36. Four Element Array with Applebaum Loops

shows a total power of 80 dB. In fact, the power at the array is only 40 dB. This difference comes from looking at the input before being processed by the four element array module instead of after processing by this module. Since both jammers are operating at 50 Hz and in phase, the amount of power appears doubled when in actuality it is not. The output spectrum in Figure (102) shows no jammer power at 50 Hz after adaptation. Again, the jammers have been totally eliminated.

4.9 Summary

The LMS algorithm implementation in BOSS was successfully tested in a test circuit, two element array, and four element array. The complex LMS algorithm implementation in BOSS was also successfully tested in a test circuit and four element complex array. The Applebaum analog algorithm implementation in BOSS started to converge to correct weight values but constantly had problems with weight divergence. After implementing the Applebaum algorithm in discrete form, simulation results closely matched theoretical predictions.

V. Conclusions and Recommendations

5.1 Conclusions

The original objective of this thesis was investigating adaptive array implemention in the Block Oriented Systems Simulator. Two different algorithms together with two different arrays were implemented in BOSS. The LMS and Applebaum algorithms were chosen and along with arrays of two and four elements. Once implemented, simulations were performed to compare simulation results to theoretical predictions. The modules implementing the algorithms were built in a fashion where they could be removed and replaced in the array systems.

The LMS algorithm proved the more successful of the two. This study showed the LMS algorithm could be implemented in BOSS using both the algorithm's real and complex form. Arrays with simple isotropic elements can be modeled in BOSS using time delays for phase shifts. For complex systems, complex exponential multipliers were used for phase shifts within the array. Putting the LMS modules into test circuits and arrays led to successful simulations that compared favorably with theoretical predictions. Array gain was steered toward sources by weight adaption and multiplication of individual element outputs. When a jammer was put in the system, the weights again adapted to produce outputs that were replicas of a desired signal while improving signal to noise ratios. Simulation convergence times agreed closely with other simulations in the literature. Null depths closely matched theoretical predictions and other simulation studies results.

The Applebaum algorithm BOSS implementation met with slightly less success. The analog implementation of the loop experienced problems with weight divergence. The weights would initially converge to theoretical values but eventually diverge. There could be some type of problem with how the Butterworth lowpass filter is implemented. The discrete implementation of this algorithm performed exactly according to theoretical predictions. When used in a sidelobe canceller circuit,

the actual weight phase and magnitude converged to theoretical values with null depths matching jammer signal powers. If a 20 dB jammer was placed at 15 degrees. a 20 db null was placed at 15 degrees after weight adaptation. When the discrete algorithm was used with a four element linear array, two jammers were nulled out with null depths matching jammer power.

Considering the objective of studying the feasability of implementing adaptive arrays in BOSS, this effort met with success. BOSS provides the necessary modules to theoretically implement most adaptive arrays. If a needed module does not exist in the BOSS library, the module can be implemented in FORTRAN and imported to BOSS. The BOSS simulation environment provides the necessary to tools to simulate arrays and adaptive algorithms. The amount of effort needed to perform the simulations was greatly reduced during this study by being able to implement needed sections in a block fashion.

5.2 Recommendations

The first recommendation concerns types of waveforms. This study used fairly simple sinusoids as the desired and jammer signals. In realistic circuits, more complex sources are used. Further simulations using possibly spread spectrum signals with tone jammers would provide more realistic results and have application to existing systems. The arrays and signal processing done in the adaptive loops could also be used as the front end of a radio type receiver.

The second recommendation involves changing the array elements. Replacing the isotropic elements with dipoles would be a next logical step in modeling arrays. The changes to the arrays would not be tremendous since dipoles are well modeled mathematically. And almost any entity that can be expressed mathematically can be implemented in BOSS.

The last recommendation involves the use of wideband jammers. With single tap LMS filters, wide band signals cannot be rejected. By placing LMS loops in

sequence to form transversal filters at each element output, wide band interference should be rejected (8:62).

Appendix A. *Appendix A - Simulation Outputs*

This appendix contains all the outputs from simulations run as a part of this thesis effort. Each section contains input and output plots for a particular BOSS circuit. The inputs are generally time plots of DC or sinusoidal signals while the output plots are time histories of weights and array output signals along with array gain patterns.

A.1 LMS Test Circuit Plots

The first test was run with the parameters in Table (1) using the circuit in Figure (29). Using a 5 Hertz sinusoid as the input to the circuit produced the output, error and weight plots along with the input plot shown in Figures (37), (38), (39),and (40).

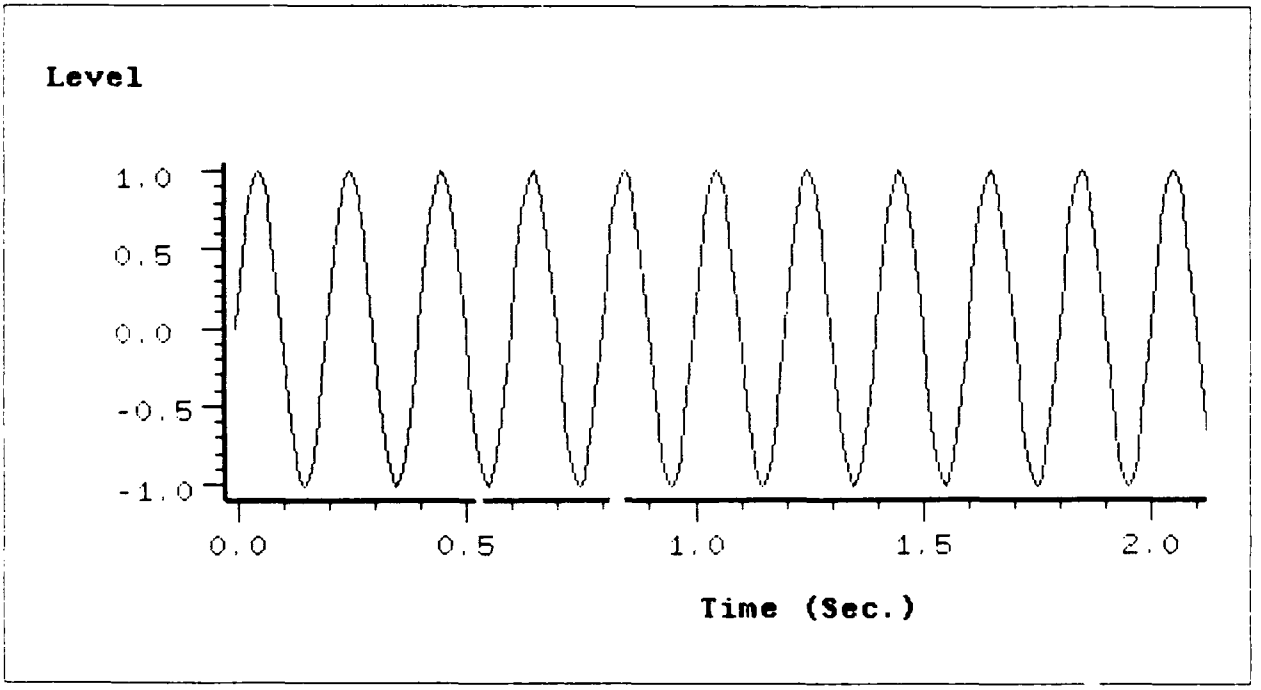

Figure 37. LMS Test Circuit 5 Hz Input with $\mu = 0.1$

```
STOP-TIME = 10.0
DT = 1.0E-2
DESIRED AMPLITUDE = 0.5
SINUSOIDAL FREQUENCY (HZ.) = 5
SINUSOIDAL PHASE (RADS.) = 0.0
SINUSOIDAL AMPLITUDE = 1.0
CONVERGENCE FACTOR  = 0.1
```

Table 1. LMS Test Circuit Parameters with $\mu = 0.1$ (5 Hz Input)

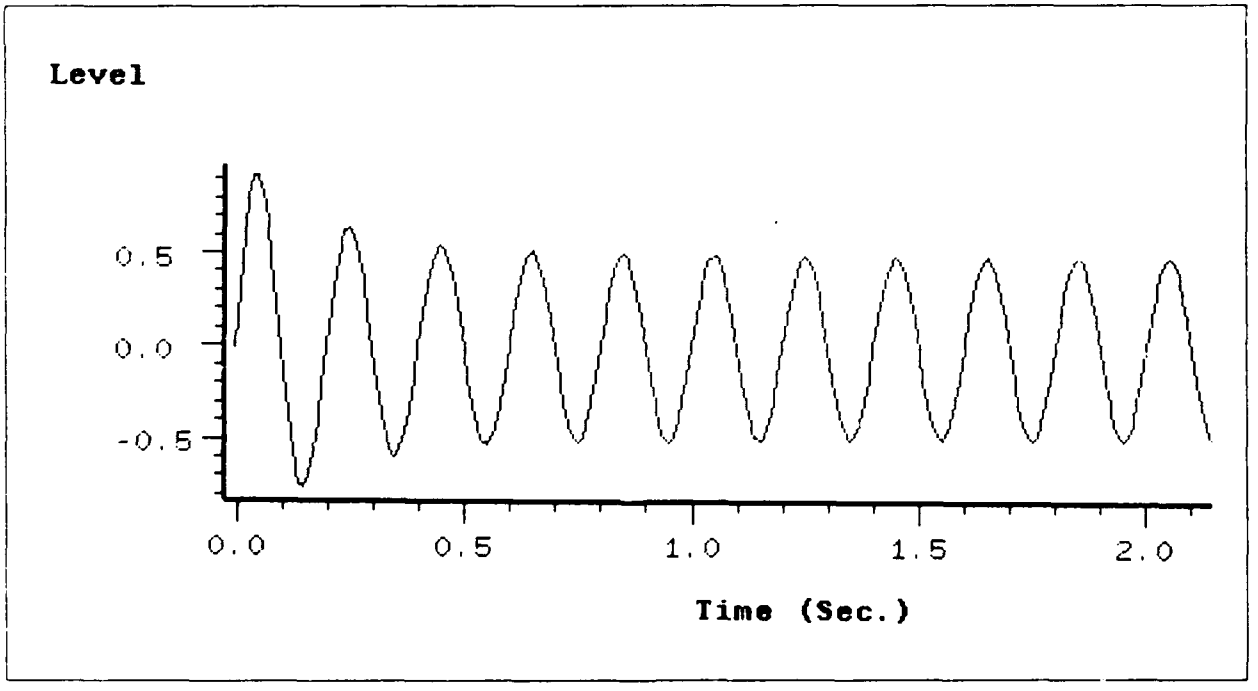

Figure 38. LMS Test Circuit 5 Hz Output with $\mu = 0.1$

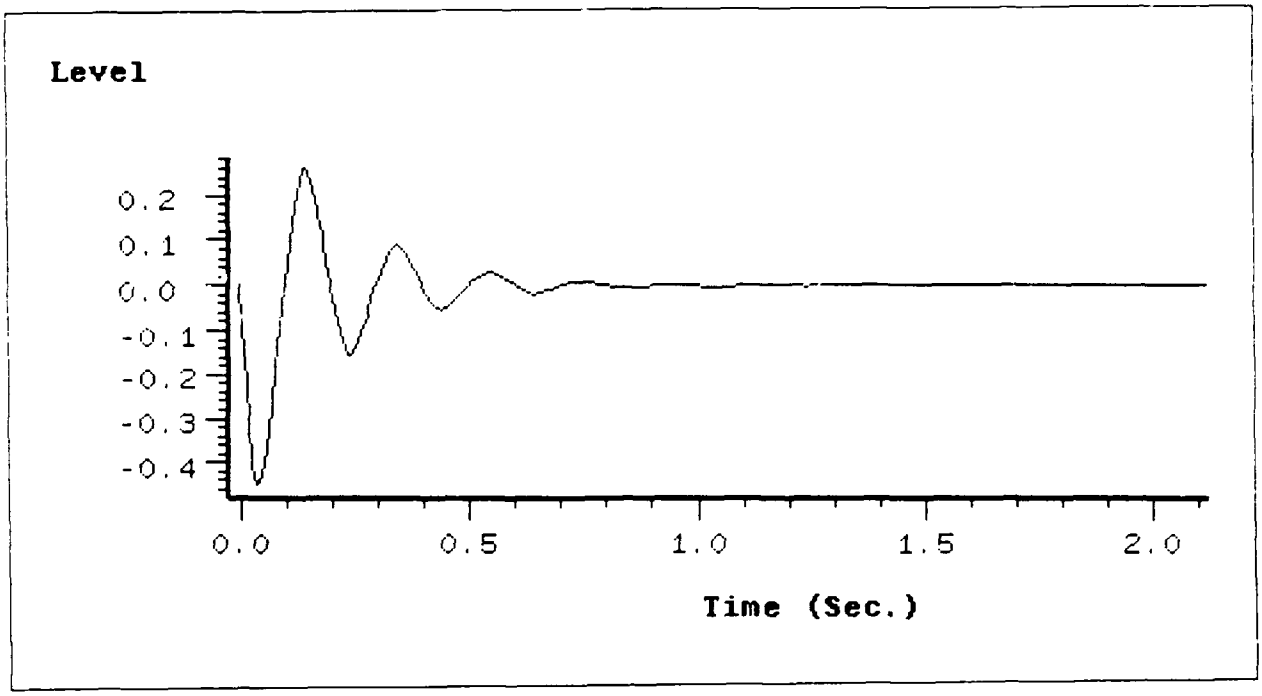

Figure 39. LMS Test Circuit Error with $\mu = 0.1$ (5 Hz Input)

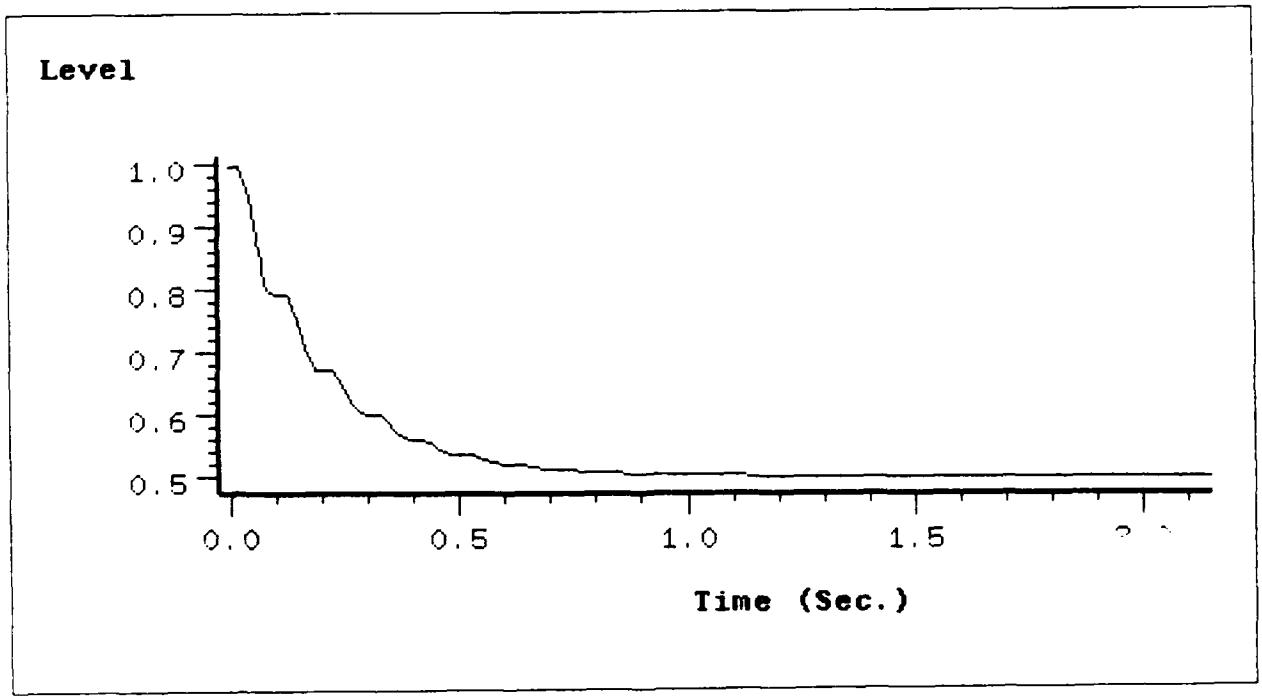

Figure 40. LMS Test Circuit Weight with $\mu = 0.1$ (5 Hz Input)

The following four figures used the same LMS circuit in Figure (29) but the input frequency was changed to 5 KHz. Table (2) reflects the parameters used for the 5 KHz simulation.

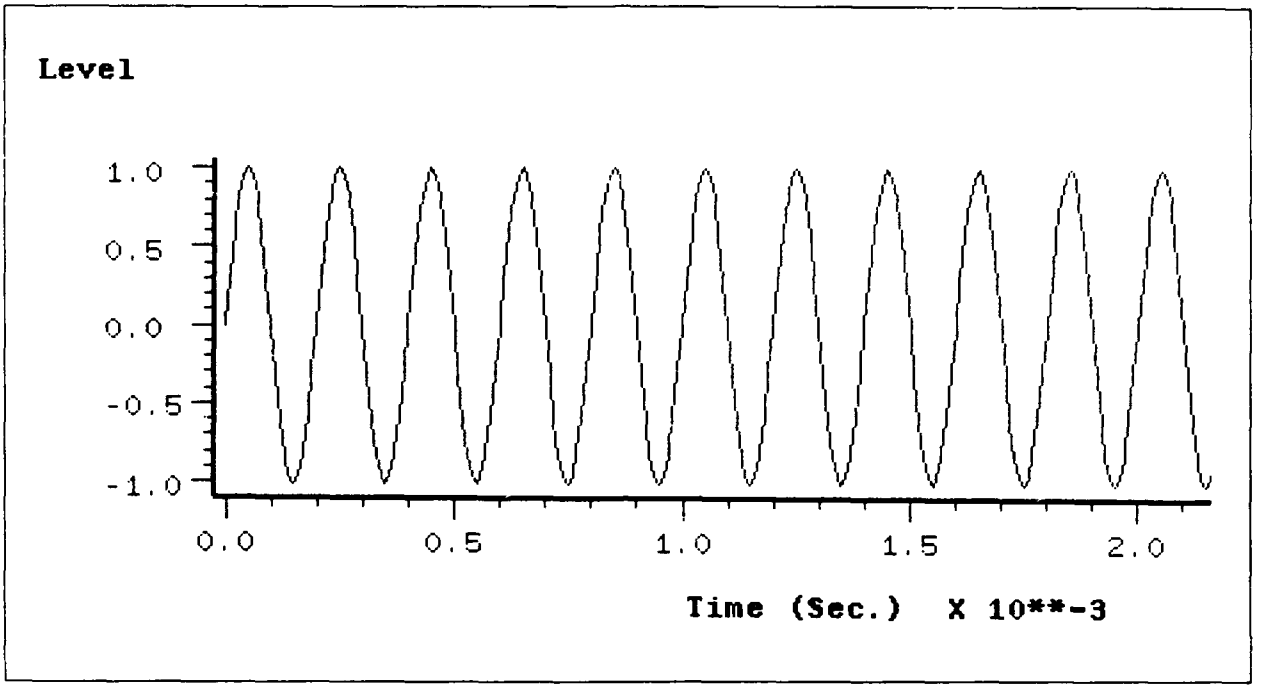

Figure 41. LMS Test Circuit 5 KHz Input with $\mu = 0.1$

LMS TEST CIRCUIT-TEST @ 5KHZ MU=0.1	
STOP-TIME =	1.0E-2
DT =	1.0e-5
DESIRED AMPLITUDE =	0.5
SINUSOIDAL FREQUENCY (HZ.) =	5000
SINUSOIDAL PHASE (RADS.) =	0.0
SINUSOIDAL AMPLITUDE =	1.0
CONVERGENCE FACTOR =	0.1

Table 2. LMS Test Circuit Parameters with $\mu = 0.1$ (5 KHz Input)

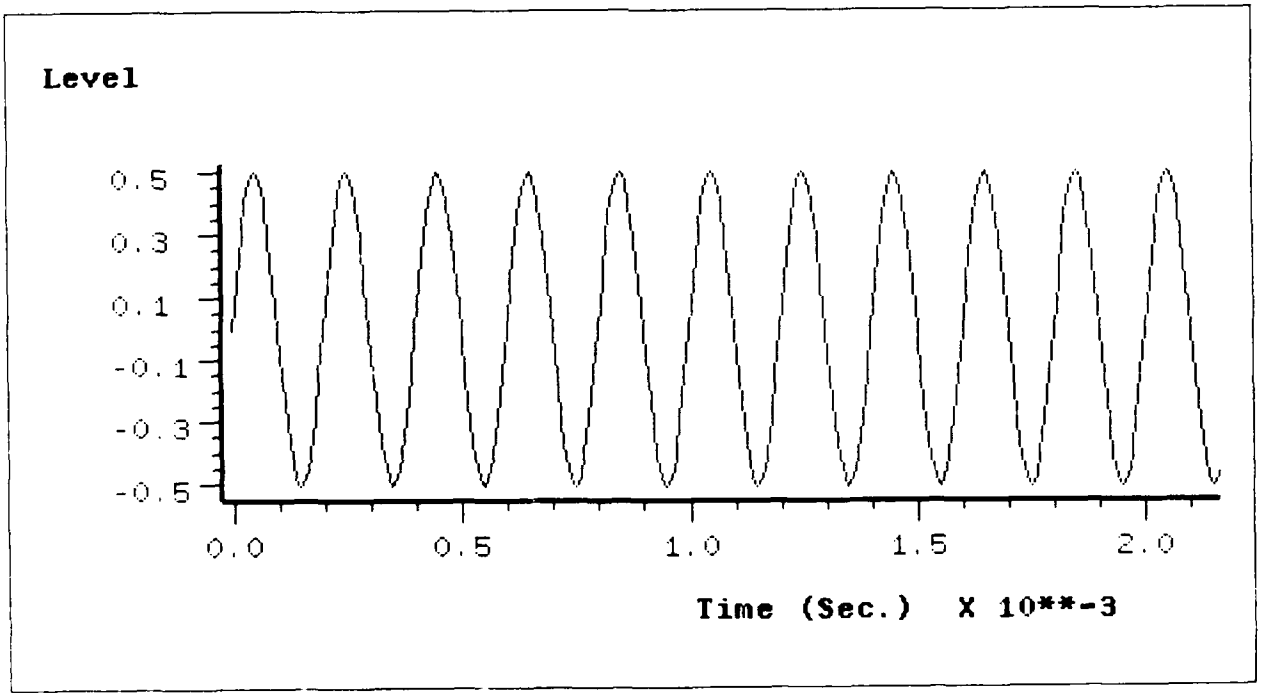

Figure 42. LMS Test Circuit 5 KHz Desired Signal with $\mu = 0.1$

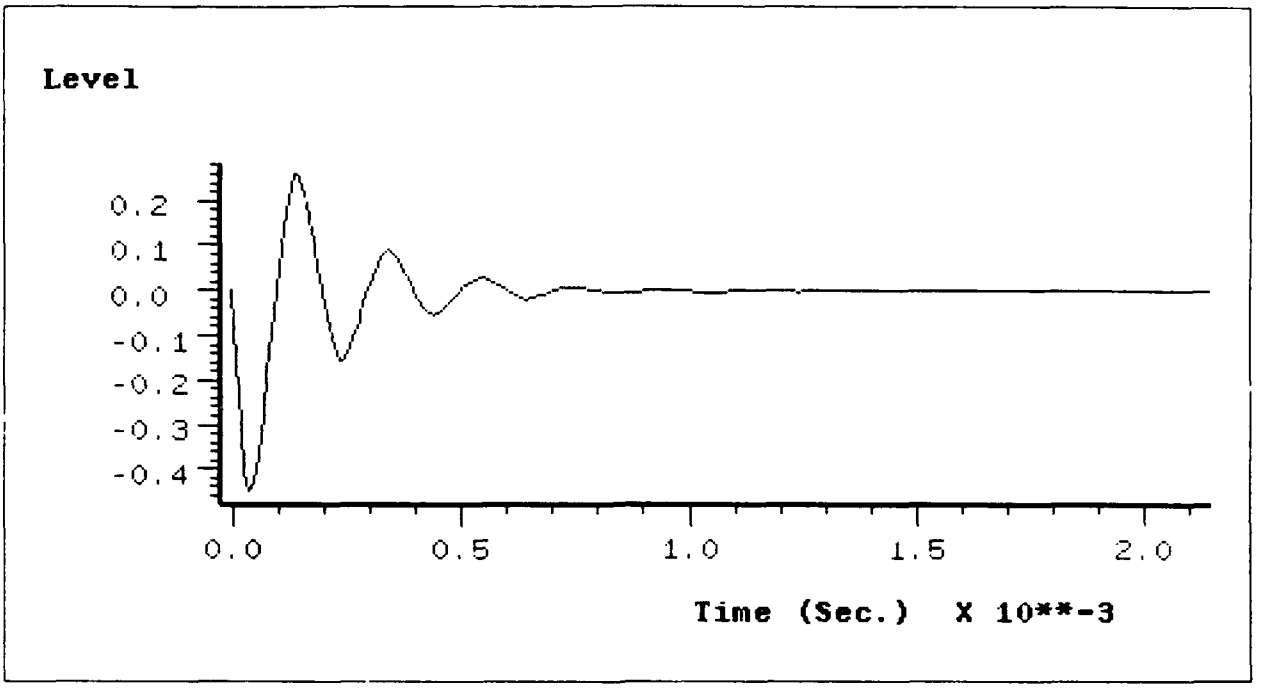

Figure 43. LMS Test Circuit Error with $\mu = 0.1$ (5 KHz Input)

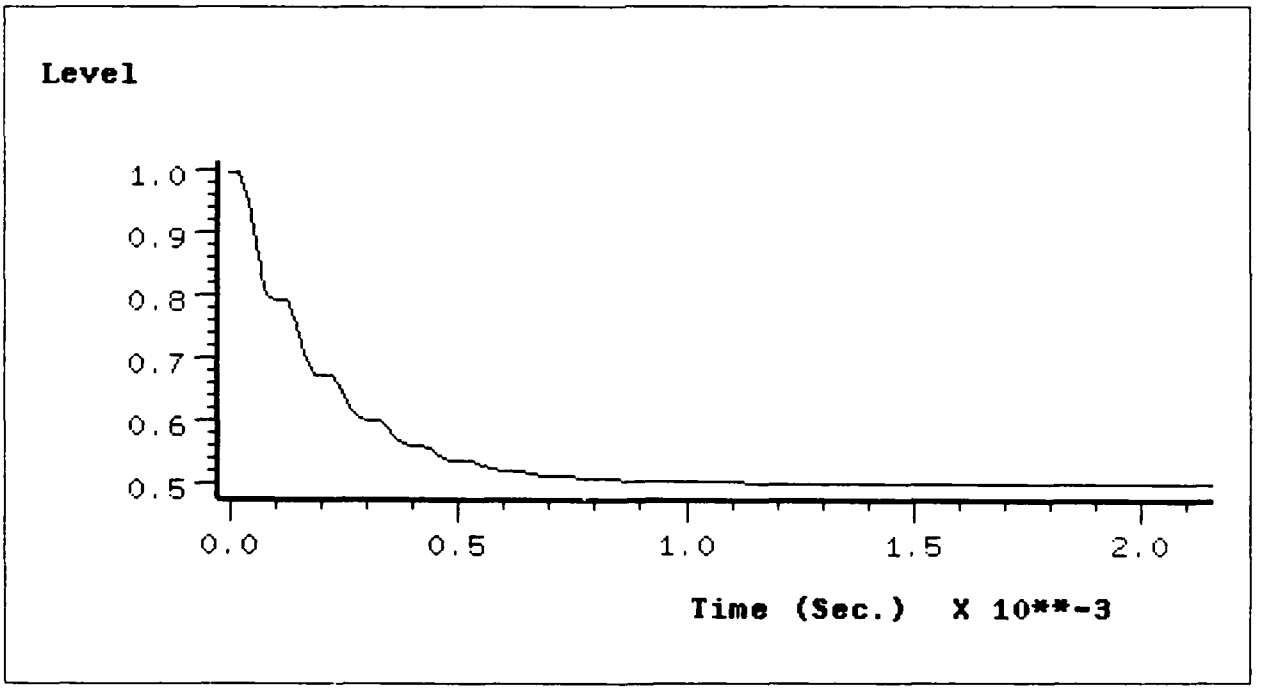

Figure 44. LMS Test Circuit Weight with $\mu = 0.1$ (5 KHz Input)

Again, the LMS test circuit was used with an input frequency of 5 MHz. The next four figures show the input, desired, error, and weight signal produced by the simulation. Table (3) reflects the parameters used for the 5 MHz simulation.

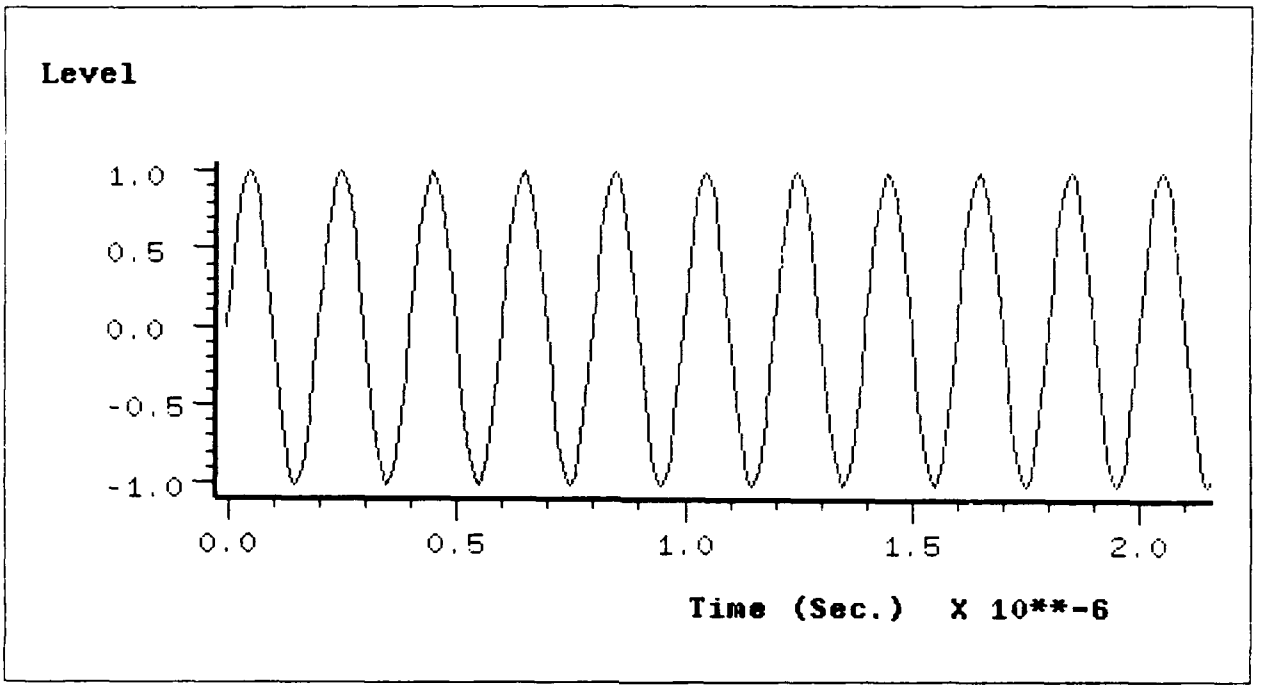

Figure 45. LMS Test Circuit 5 MHz Input with $\mu = 0.1$

Menu **LMS TEST CIRCUIT-TEST AT 5MHZ MU=0.1**

STOP-TIME = 1.0e-5
DT = 1.0e-8
DESIRED AMPLITUDE = 0.5
SINUSOIDAL FREQUENCY (HZ.) = 5000000
SINUSOIDAL PHASE (RADS.) = 0.0
SINUSOIDAL AMPLITUDE = 1.0
CONVERGENCE FACTOR = 0.1

Table 3. LMS Test Circuit Parameters with $\mu = 0.1$ (5 MHz Input)

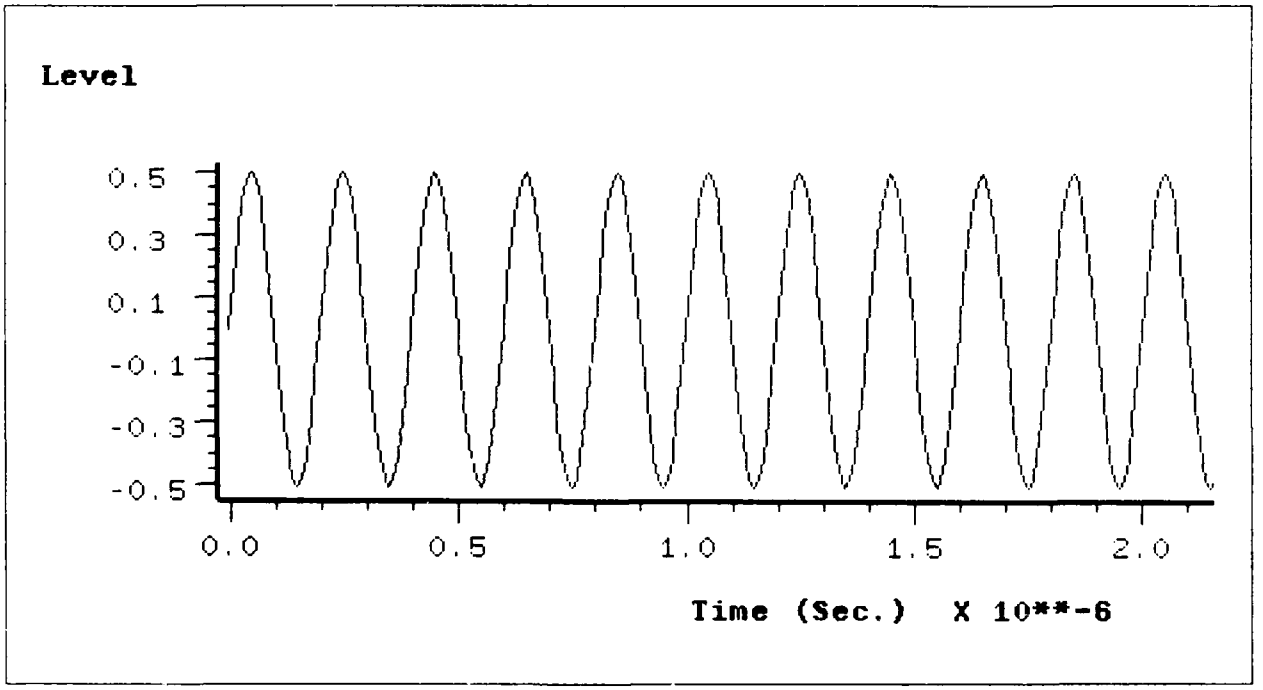

Figure 46. LMS Test Circuit 5 MHz Desired Signal with $\mu = 0.1$

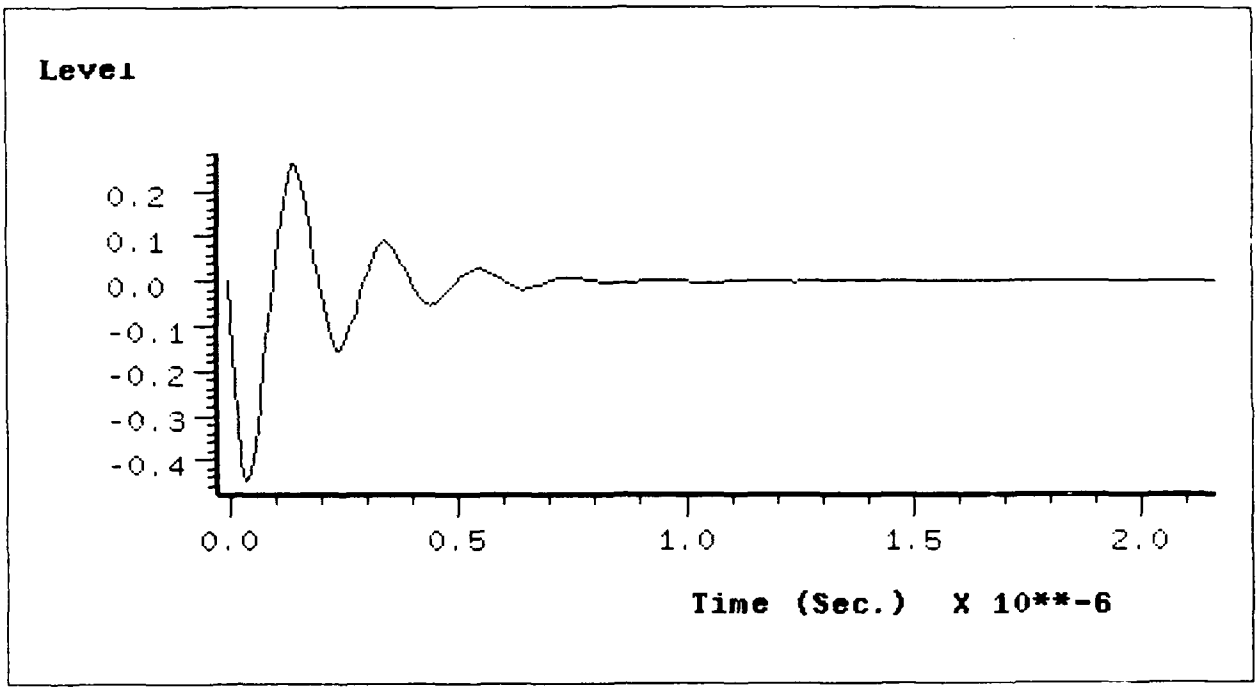

Figure 47. LMS Test Circuit Error with $\mu = 0.1$ // (5 MHz Input)

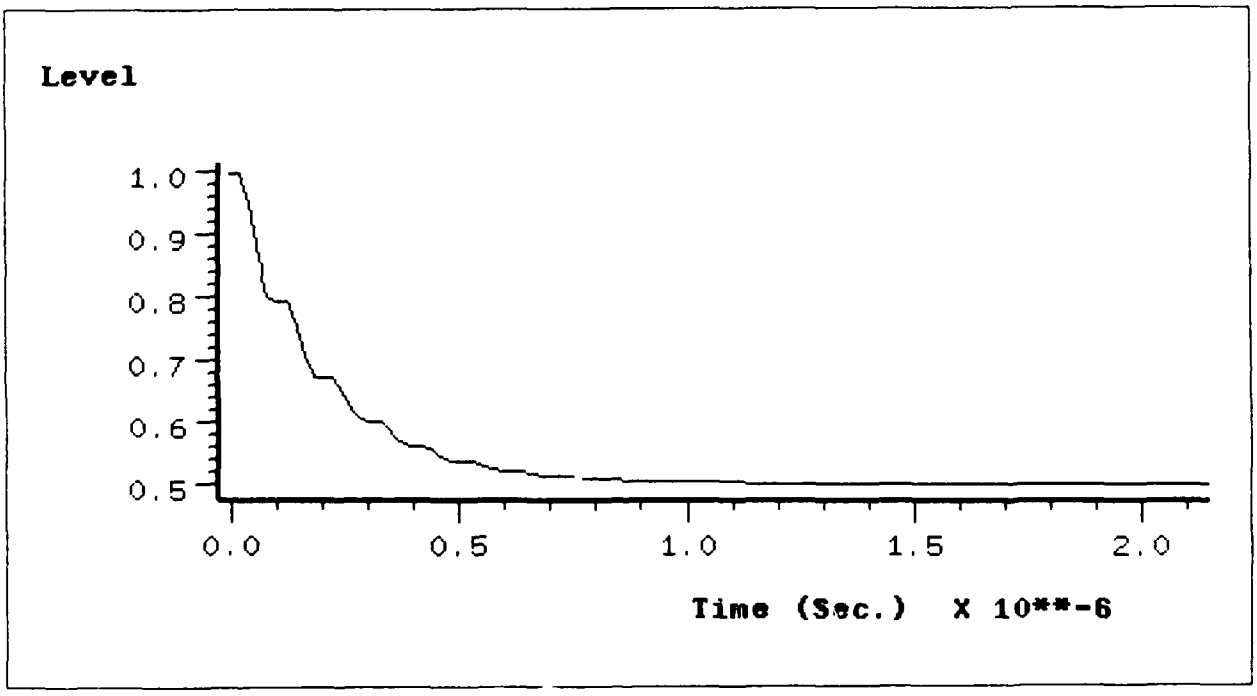

Figure 48. LMS Test Circuit Weight with $\mu = 0.1$ // (5 MHz Input)

To further test the circuit, the input frequency was changed to 5 GHz. It was necessary to find any frequency dependencies in the circuit that might occur as a result of BOSS or digital simulations in general. The circuit itself is composed of simple adders, multipliers, and delays. There were no frequency dependent components designed in the system. If there are no frequency dependencies, there is no reason to not perform these simulations at 5 Hz if the results are directly translatable to 5 MHz or 5 GHz. The parameters used for the simulation at 5 GHz are in Table (4). The convergence factor μ was again set to 0.1. The results are shown in the next four figures.

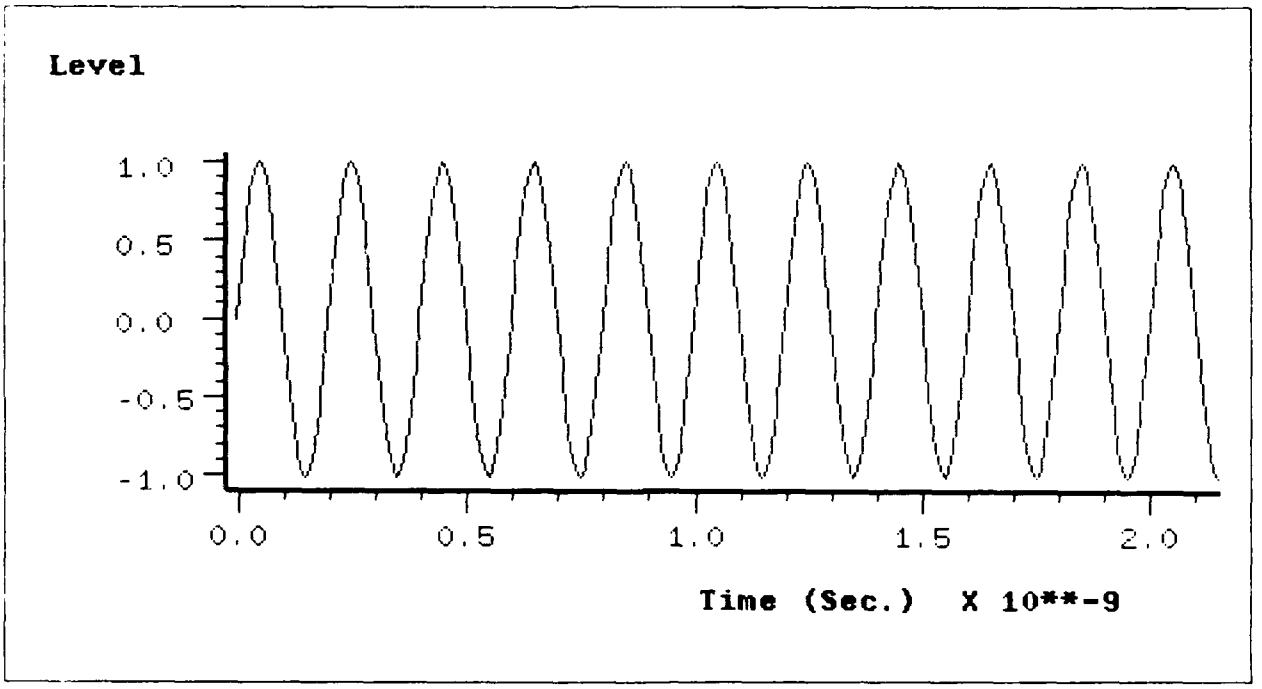

Figure 49. LMS Test Circuit 5 GHz Input with $\mu = 0.1$

LMS TEST CIRCUIT-TEST @ 5GHZ MU=0.1

STOP-TIME = 1.0e-8
DT = 1.0e-11
DESIRED AMPLITUDE = 0.5
SINUSOIDAL FREQUENCY (HZ.) = 5000000000
SINUSOIDAL PHASE (RADS.) = 0.0
SINUSOIDAL AMPLITUDE = 1.0
CONVERGENCE FACTOR = 0.1

Table 4. LMS Test Circuit Parameters with $\mu = 0.1$ (5 GHz Input)

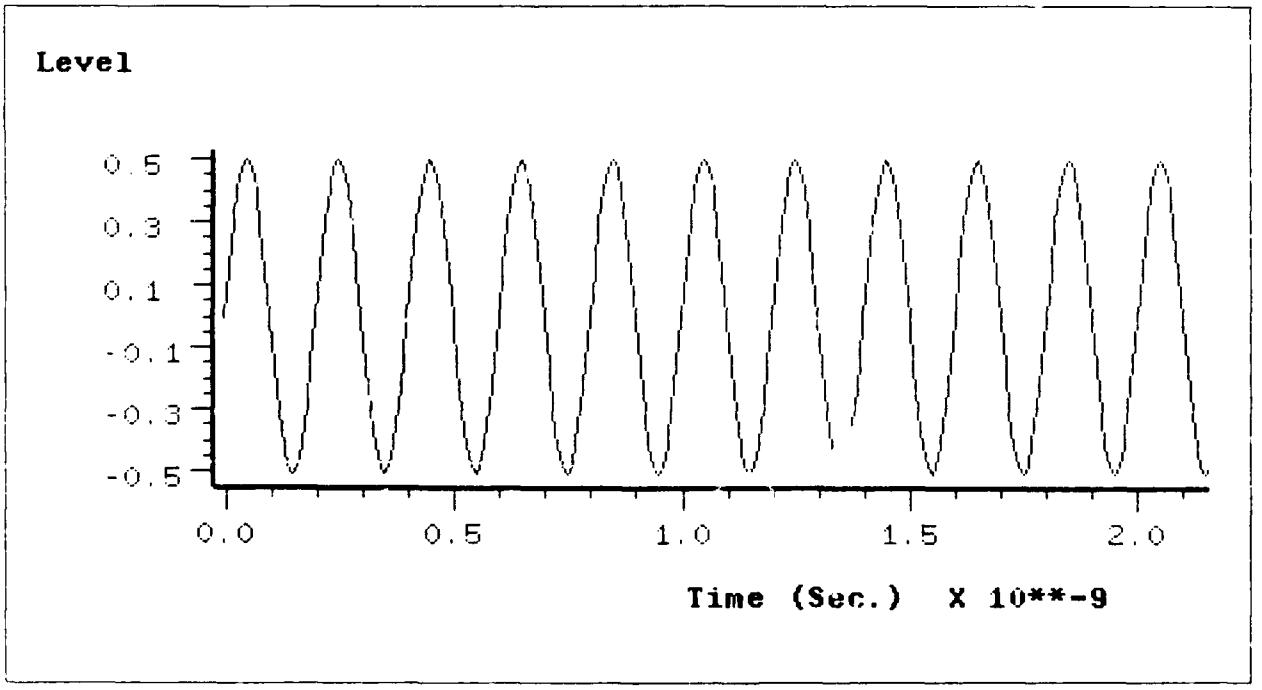

Figure 50. LMS Test Circuit 5 GHz Desired Signal with $\mu = 0.1$

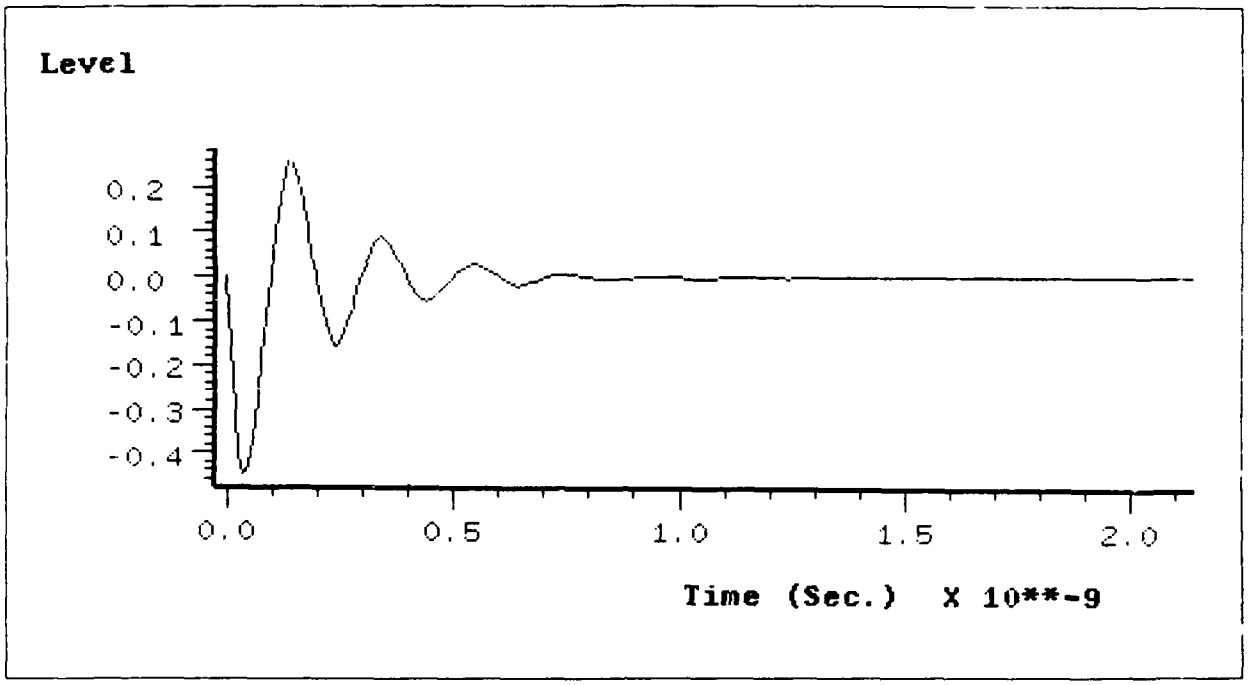

Figure 51. LMS Test Circuit Error with $\mu = 0.1$ (5 GHz Input)

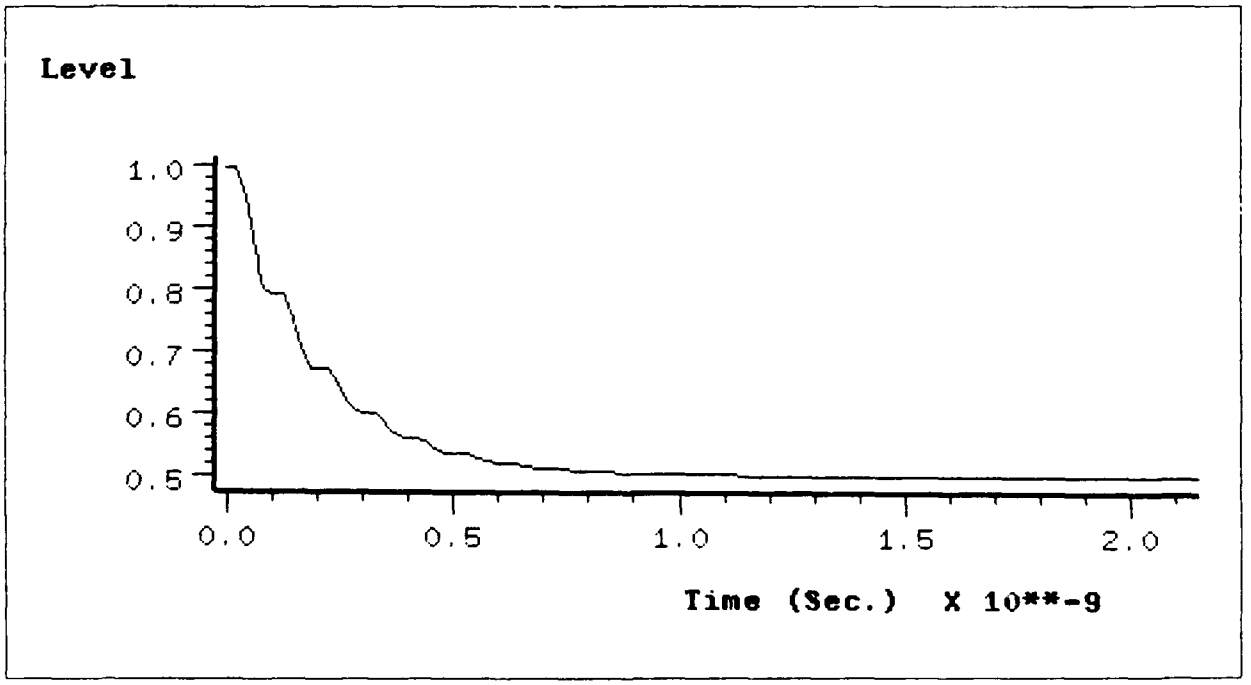

Figure 52. LMS Test Circuit Weight with $\mu = 0.1$ (5 GHz Input)

The convergence factor, μ, was varied during the next section to observe the effect μ had on the circuit. As can be seen in Table (5), the convergence factor was reduced to 0.05. The output, error, and weight signals are shown in the next three figures for the 5 Hz input shown in Figure (37).

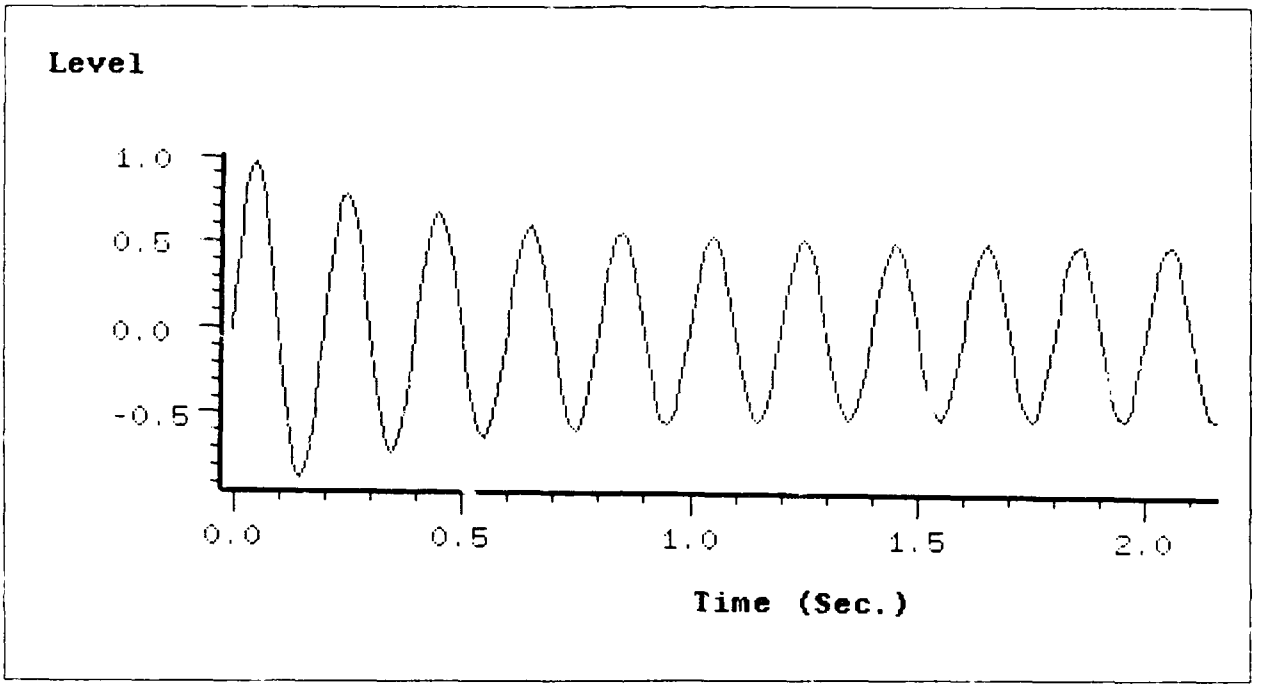

Figure 53. LMS Test Circuit 5 Hz Output with $\mu = 0.05$

Menu LMS TEST CIRCUIT-TEST @ 5 HZ MU=0.05

STOP-TIME = 10.0
DT = 1.0E-2
DESIRED AMPLITUDE = 0.5
SINUSOIDAL FREQUENCY (HZ.) = 5
SINUSOIDAL PHASE (RADS.) = 0.0
SINUSOIDAL AMPLITUDE = 1.0
CONVERGENCE FACTOR = 5.0E-2

Table 5. LMS Test Circuit Parameters with $\mu = 0.05$ (5 Hz Input)

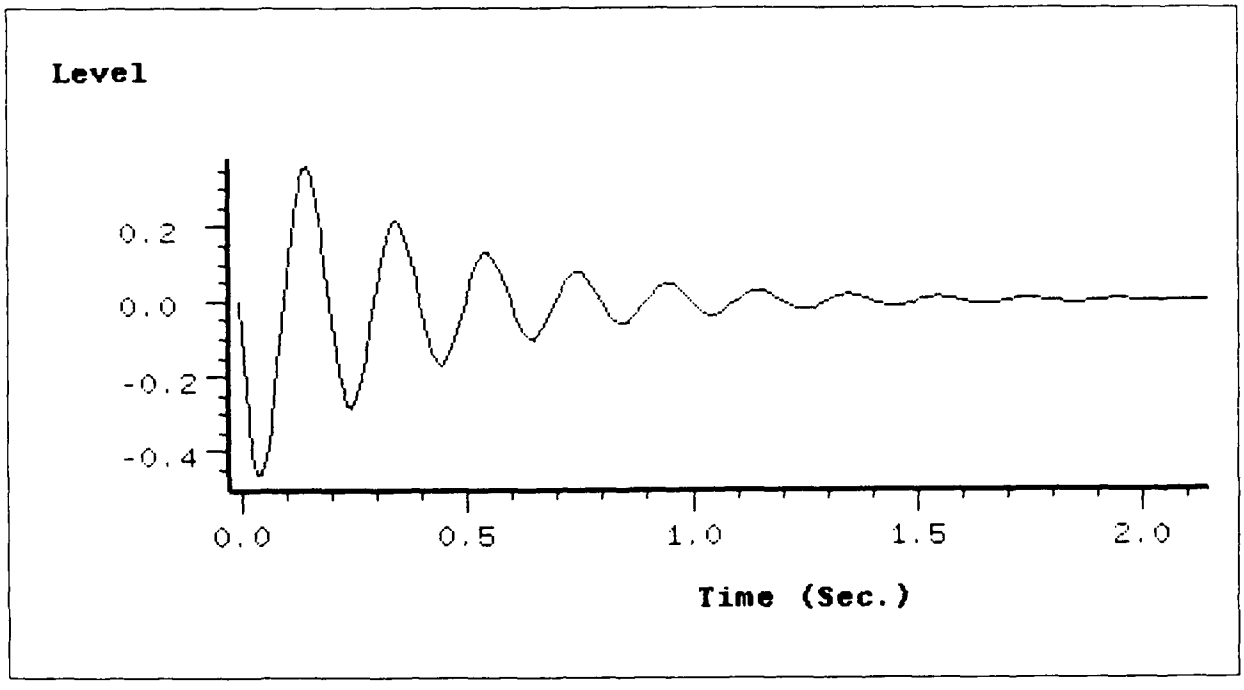

Figure 54. LMS Test Circuit Error with $\mu = 0.05$ (5 Hz Input)

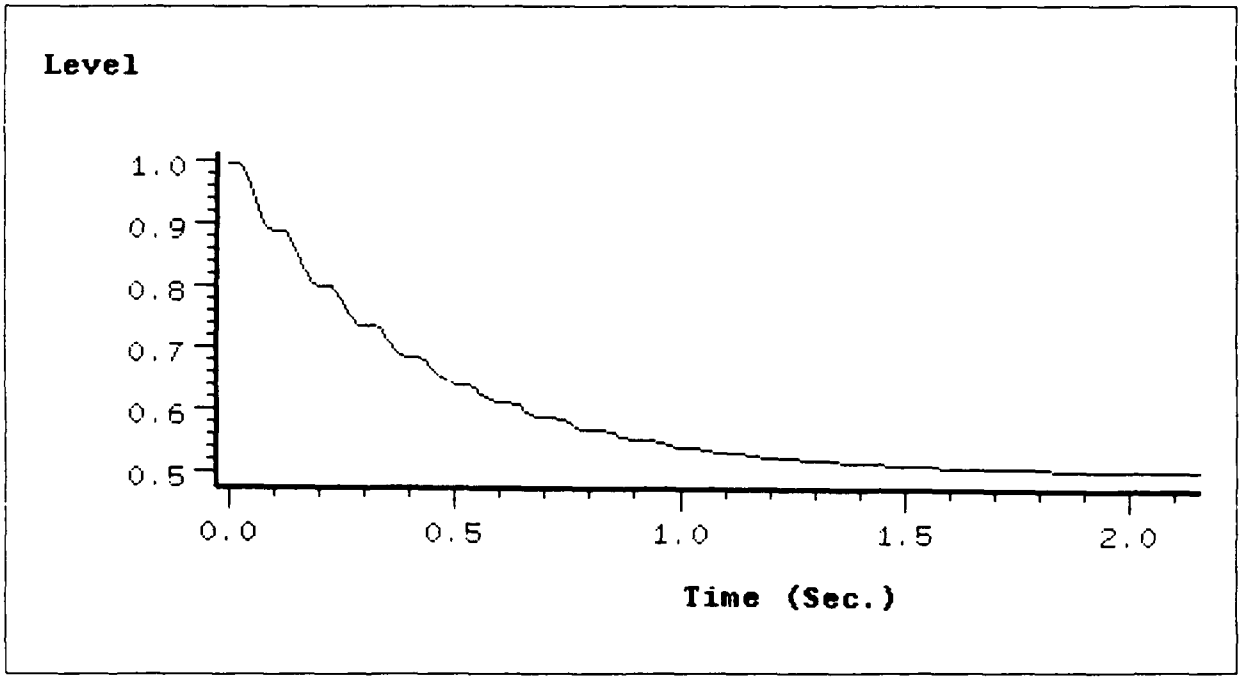

Figure 55. LMS Test Circuit Weight with $\mu = 0.05$ (5 Hz Input)

The next series of figures show the results of increasing the convergence factor to 0.5. As the convergence factor increased, the time of convergence decreased. Table (6) contains the parameters used for the simulation with μ set to 0.5.

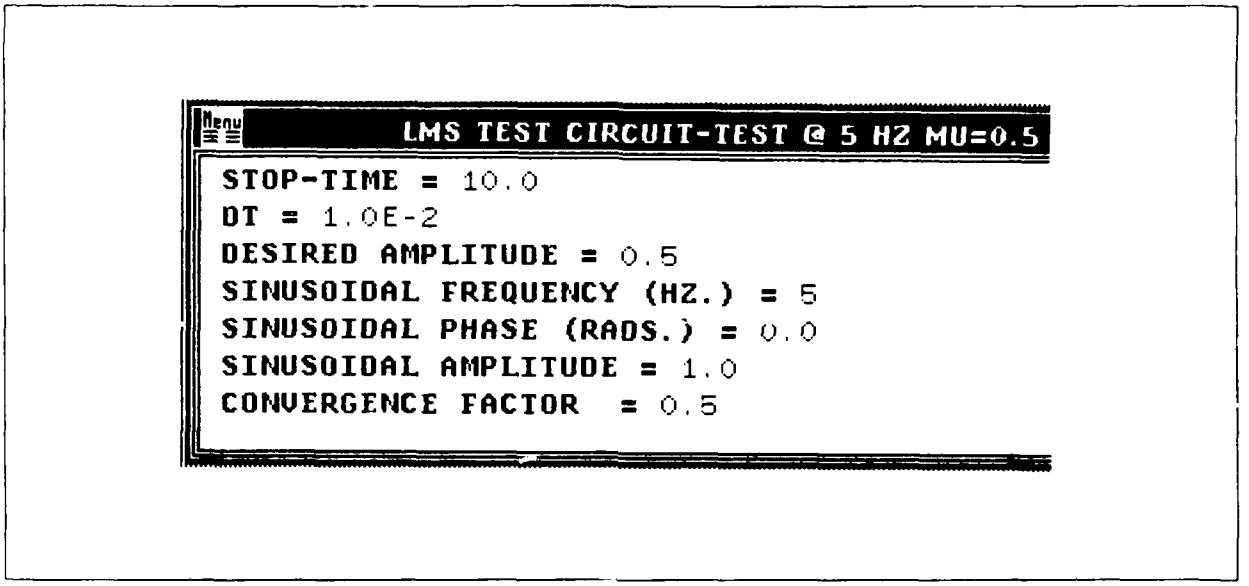

Table 6. LMS Test Circuit Parameters with $\mu = 0.5$ (5 Hz Input)

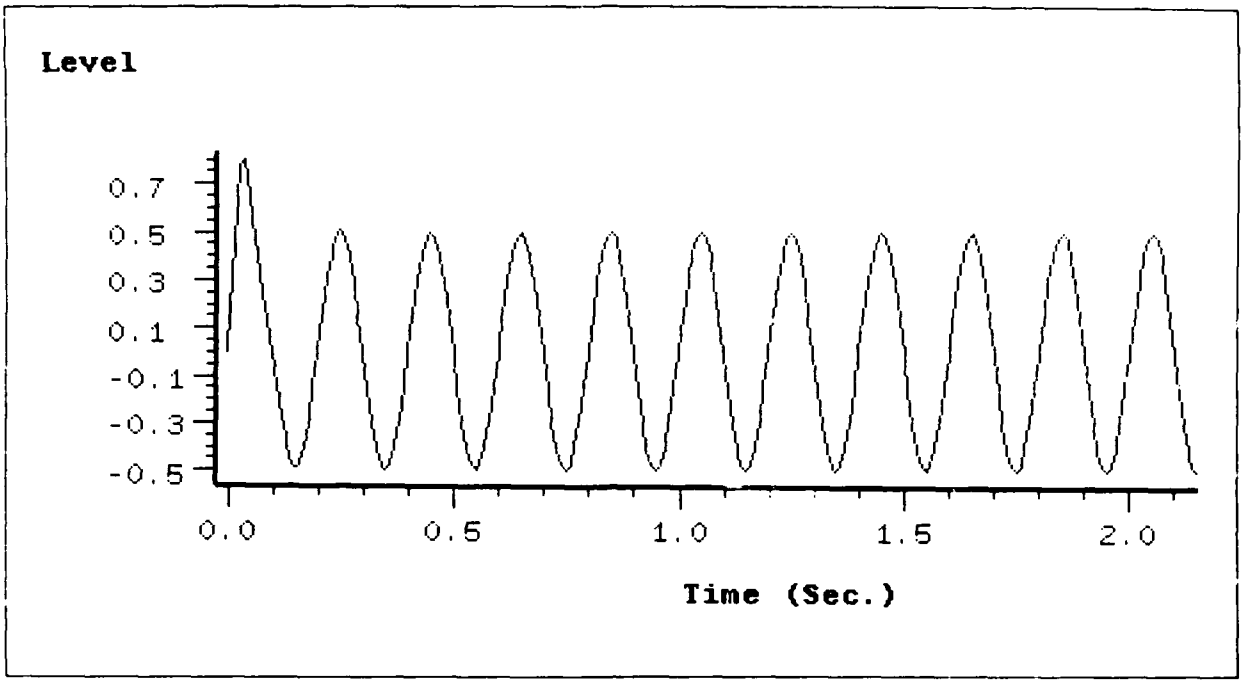

Figure 56. LMS Test Circuit 5 Hz Output with $\mu = 0.5$

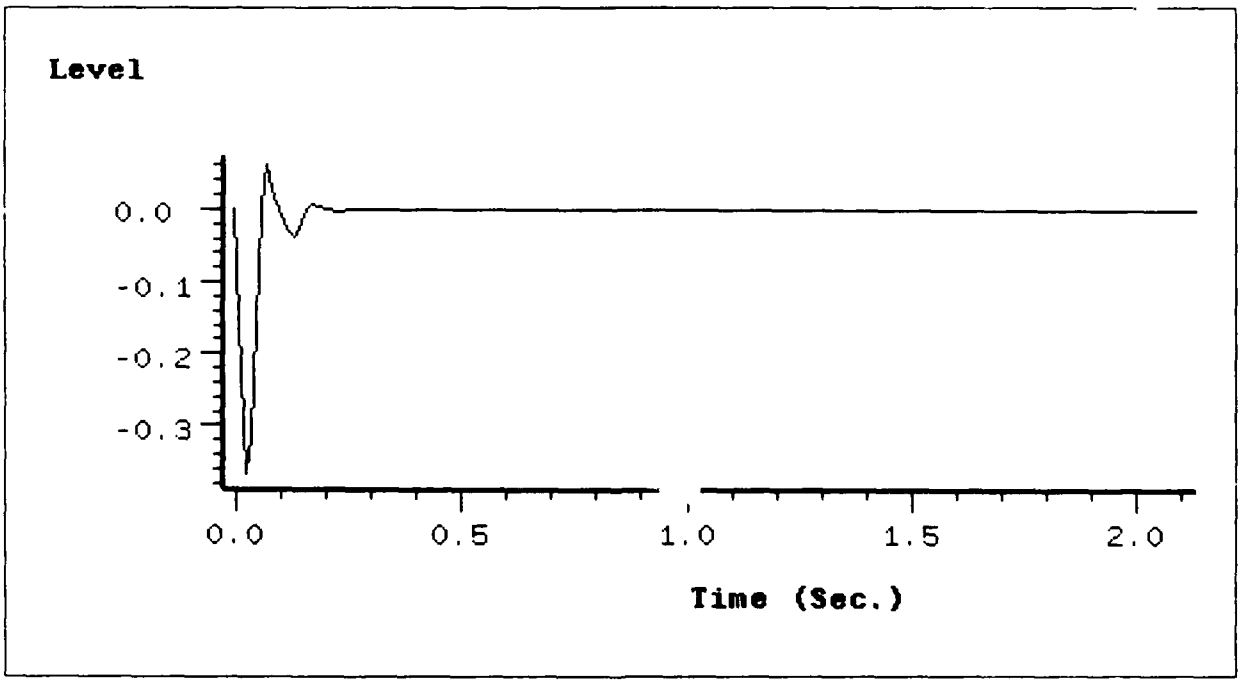

Figure 57. LMS Test Circuit Error with $\mu = 0.5$ (5 Hz Input)

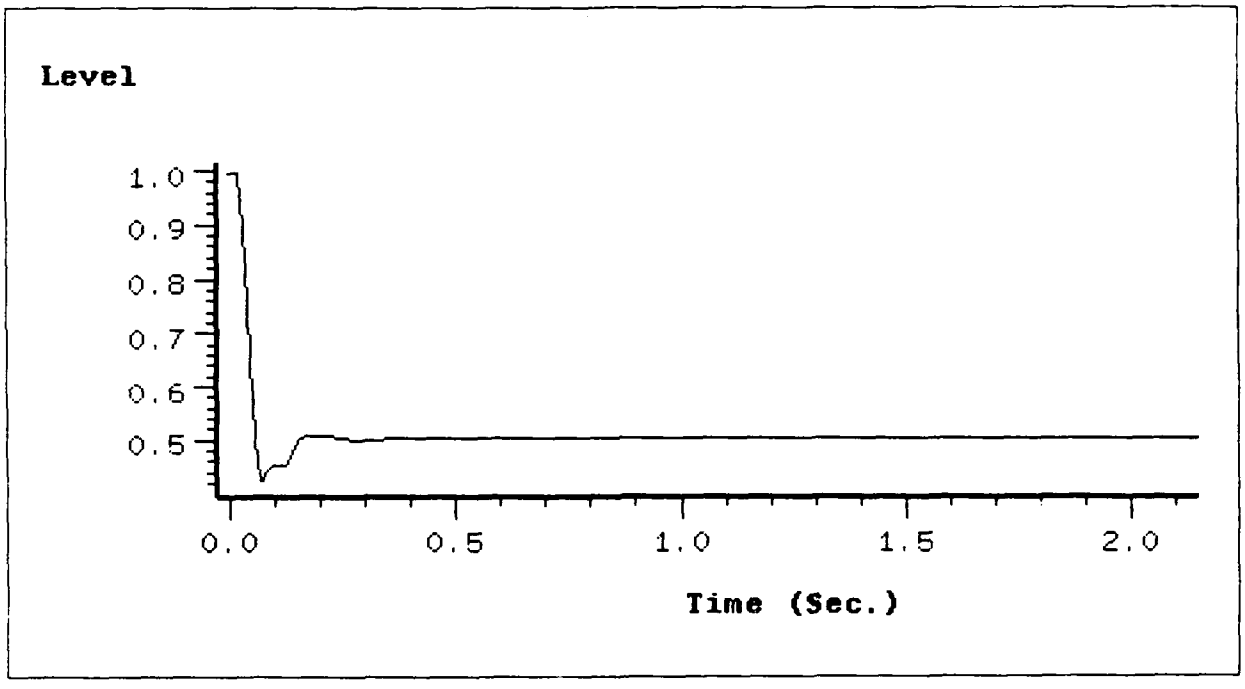

Figure 58. LMS Test Circuit Weight with $\mu = 0.5$ (5 Hz Input)

A.2 Two Element Array Simulation Results

The following table and three figures are for the first simulation using the two element array system in Figure (31). There was one transmitter with the tone jammer amplitude reduced to near zero. The noise source's power was also reduced to near zero. The transmitter was placed broadside to the array so there was no relative phase shift between elements.

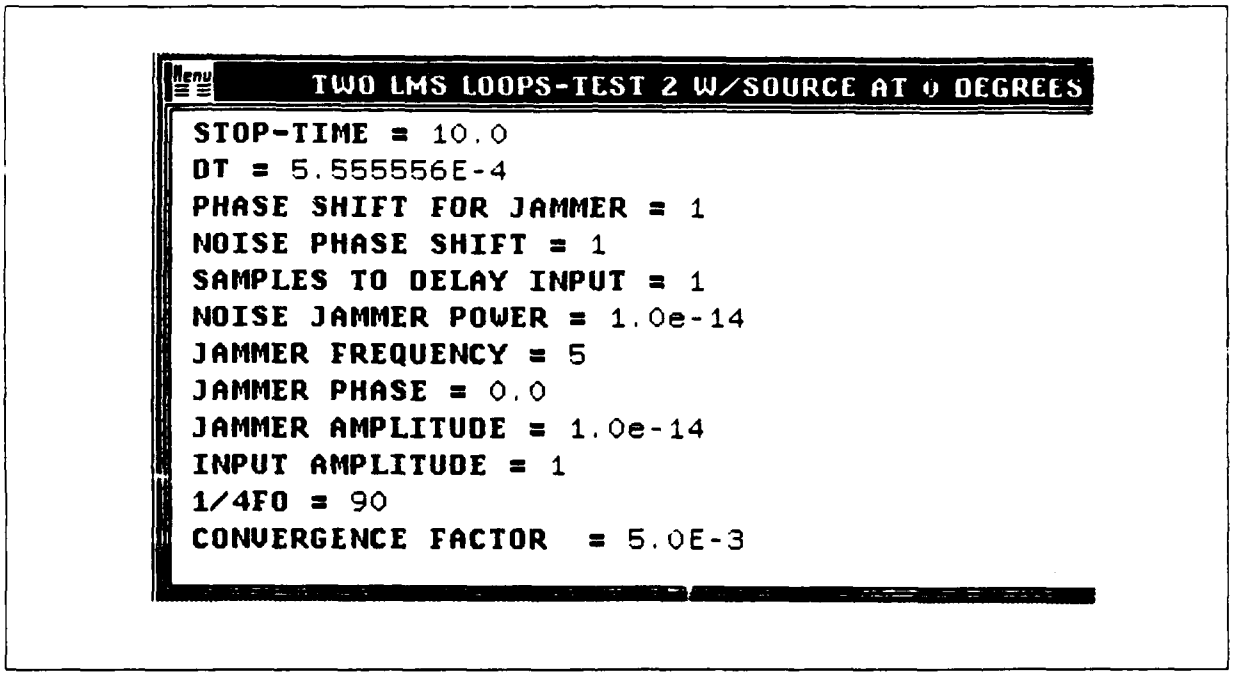

Table 7. Parameters for Two Element Array with Source at 0 Degrees

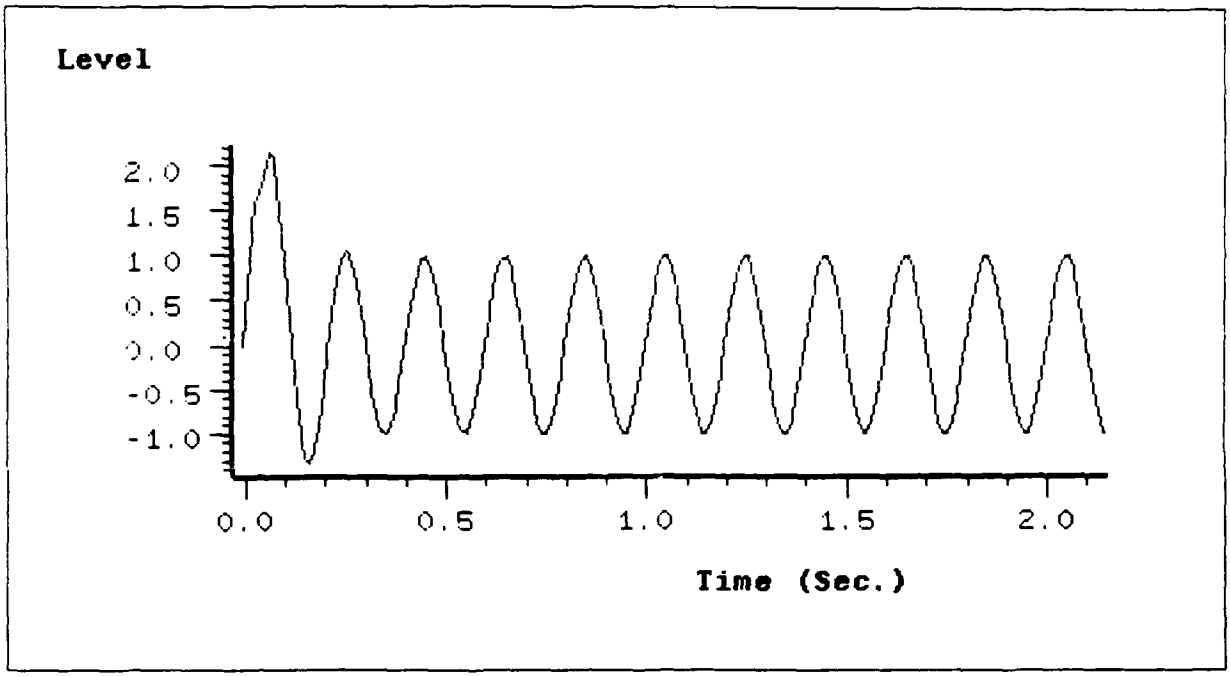

Figure 59. Two Element Array Output with Source at 0 Degrees

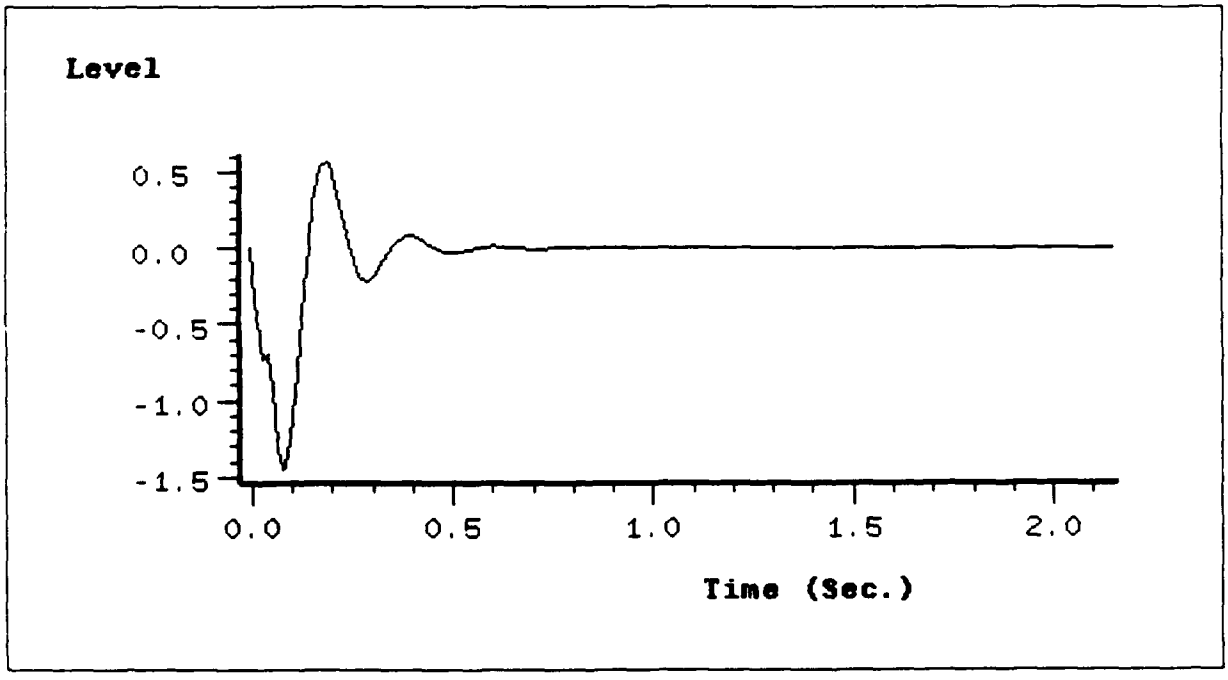

Figure 60. Two Element Array Error with Source at 0 Degrees

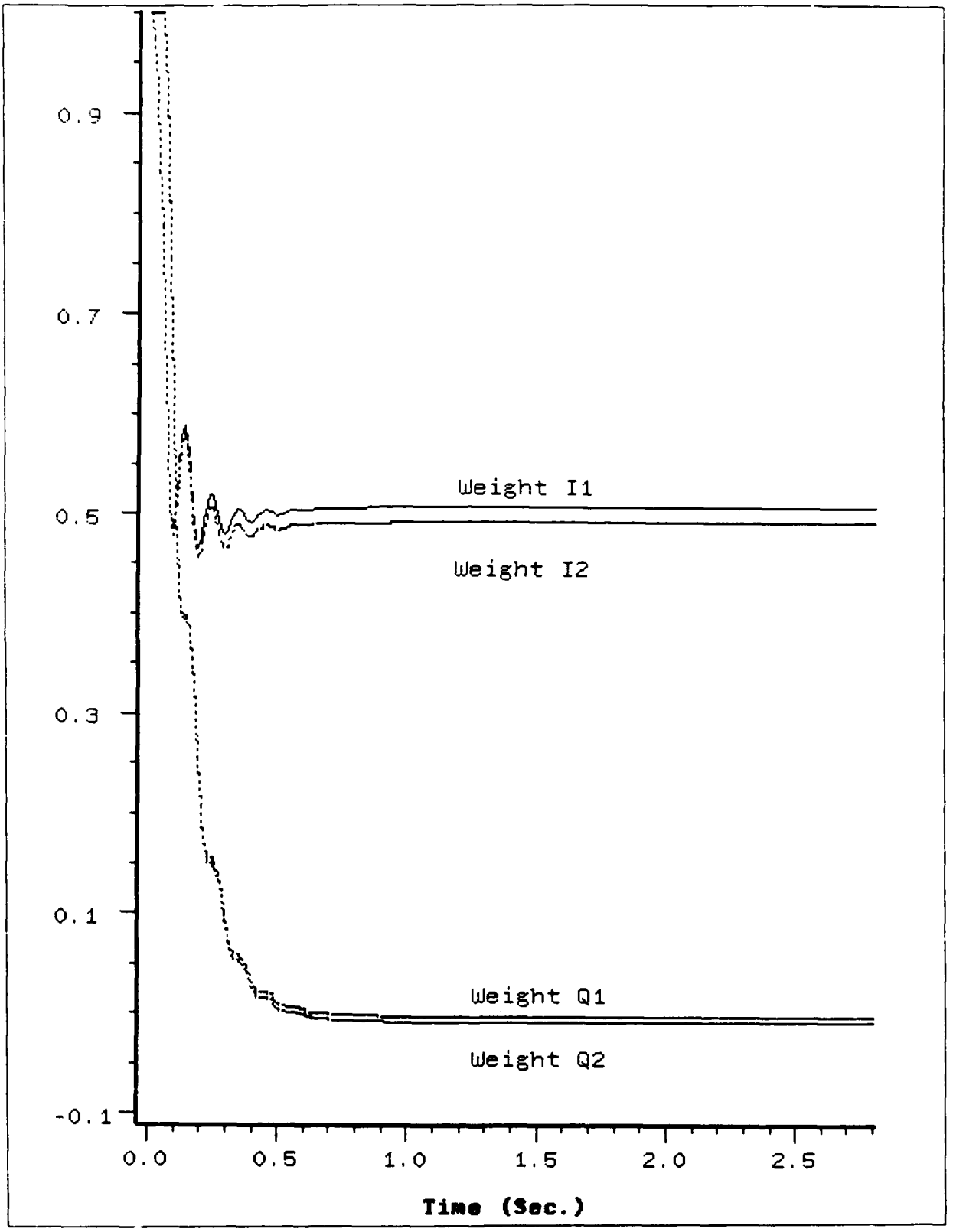

Figure 61. Two Element Array Weights with Source at 0 Degrees

The next simulation involved a source at 30 degrees relative to t array. The simulation parameters are in Table (8). The jammer and noise powers were again kept near zero. The plots for the output and error signals were very similar to Figures (59) and (60) and are not duplicated here. Only the weight plot is shown.

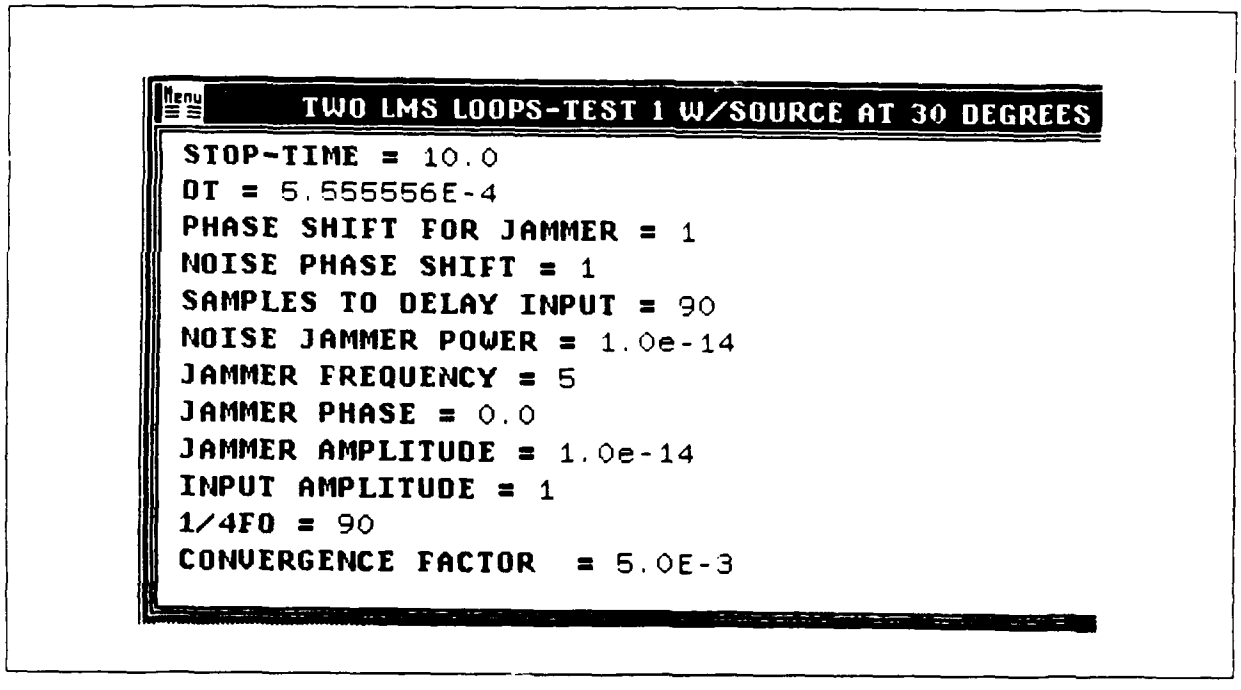

Table 8. Parameters for Two Element Array with Source at 30 Degrees

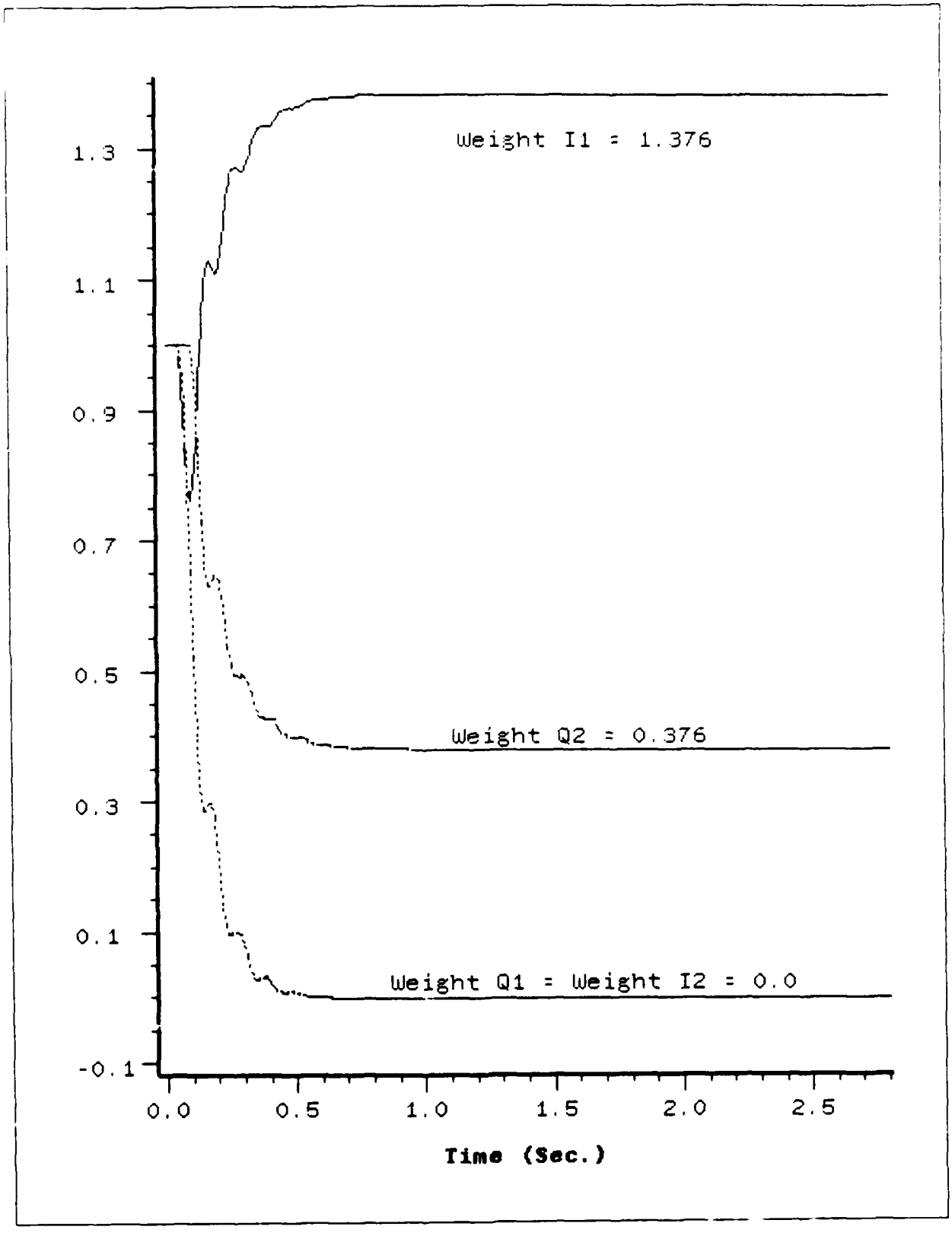

Figure 62. Two Element Array Weights with Source at 30 Degrees

The final test of the two element array LMS system involved inclusion of a jammer. The jammer amplitude and phase along with the other simulation parameters are shown in Table (9). Also included here are the plots for the output and error signals as well as the weights.

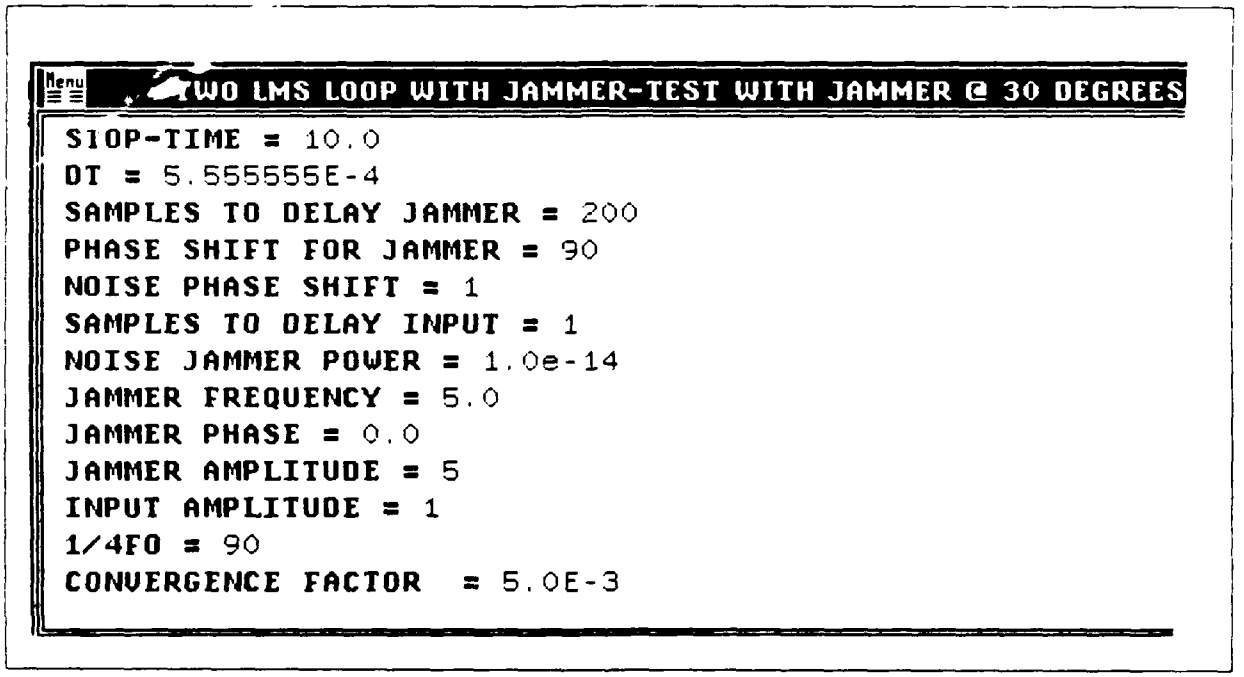

Table 9. Parameters for Two Element Array with Source at 0 Degrees and Jammer at 30 degrees

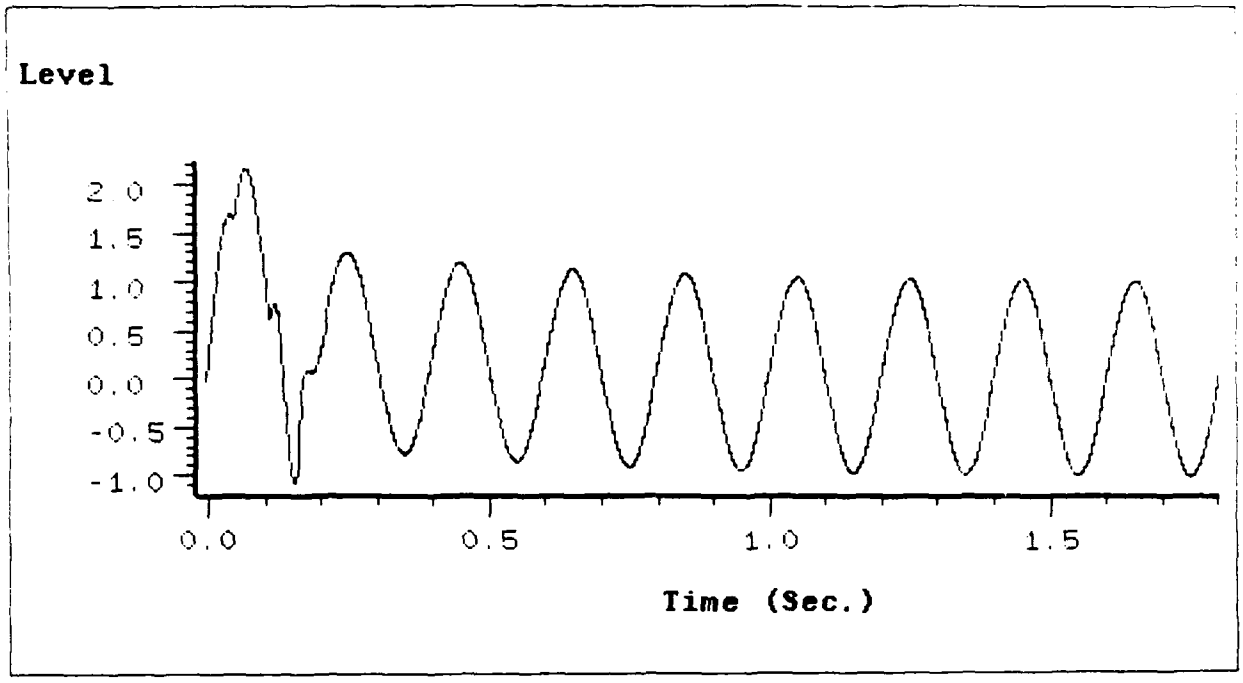

Figure 63. Two Element Array Output with Source at 0 Degrees and Jammer at 30 Degrees

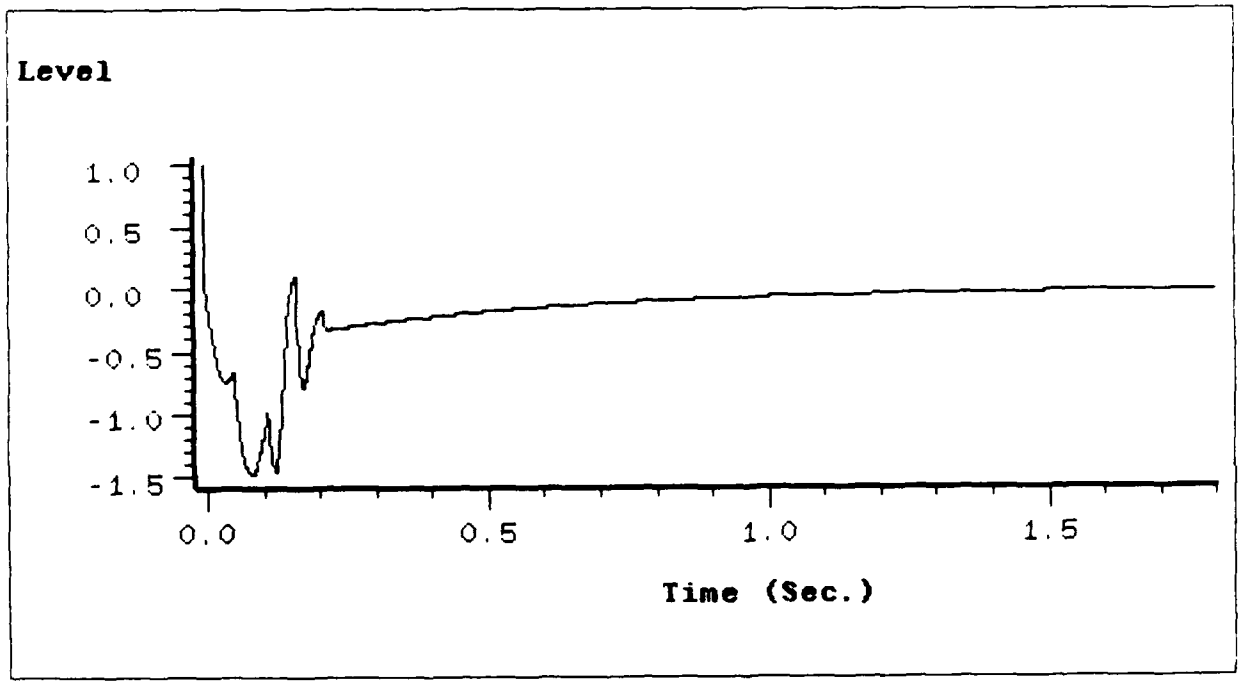

Figure 64. Two Element Array Error with Source at 0 Degrees and Jammer at 30 Degrees

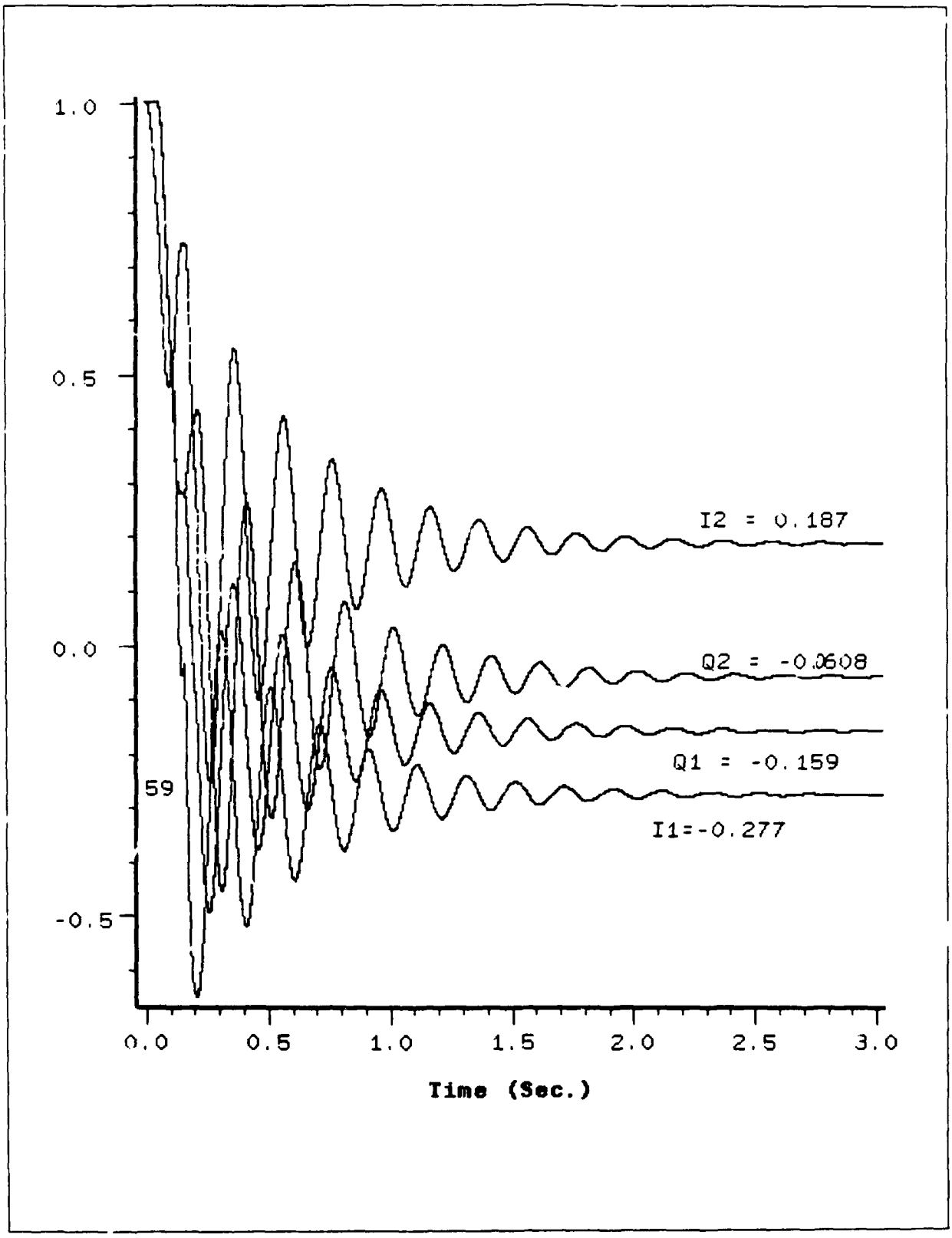

Figure 65. Two Element Array Weights with Source at 0 Degrees and Jammer at 30 Degrees

A.3 Four Element Array Simulation Results

The simulation steps for the four element array system were the same as for the two element system. The system used a square geometry with the array elements spaced half a wavelength apart. The source was originally placed at 0 degrees with reference to two of the elements, moved to 30 degrees, and returned to 0 degrees when a jammer as added at 30 degrees. The parameter listings and plots for the output, error, and weight signals are contained in the following three tables and seven figures.

```
FOUR LMS LOOPS-TEST 1 W/SOURCE AT 0 DEGREES
STOP-TIME = 10.0
DT = 5.555556E-4
DESIRED FREQ (HZ) = 5.0
DESIRED PHASE (RADS) = 0.0
DESIRED AMPLITUDE = 1.0
1/4FO = 90
CONVERGENCE FACTOR  = 5.0E-3
SIGNAL DELAY FOR ELEMENT 2 = 1
SIGNAL DELAY FOR ELEMENT 3 = 180
SIGNAL DELAY FOR ELEMENT 4 = 180
JAMMER DELAY FOR ELEMENT 4 = 180
JAMMER DELAY FOR ELEMENT 3 = 180
JAMMER DELAY FOR ELEMENT 2 = 1
NOISE DELAY FOR ELEMENT 4 = 180
NOISE DELAY FOR ELEMENT 3 = 180
NOISE DELAY FOR ELEMENT 2 = 1
NOISE POWER = 1.0e-14
JAMMER FREQ (HZ) = 5.0
JAMMER AMPLITUDE = 1.0e-14
INPUT SIGNAL FREQ (HZ) = 5.0
INPUT SIGNAL AMPLITUDE = 1.0
```

Table 10. Parameters for Four Element Array with Source at 0 Degrees

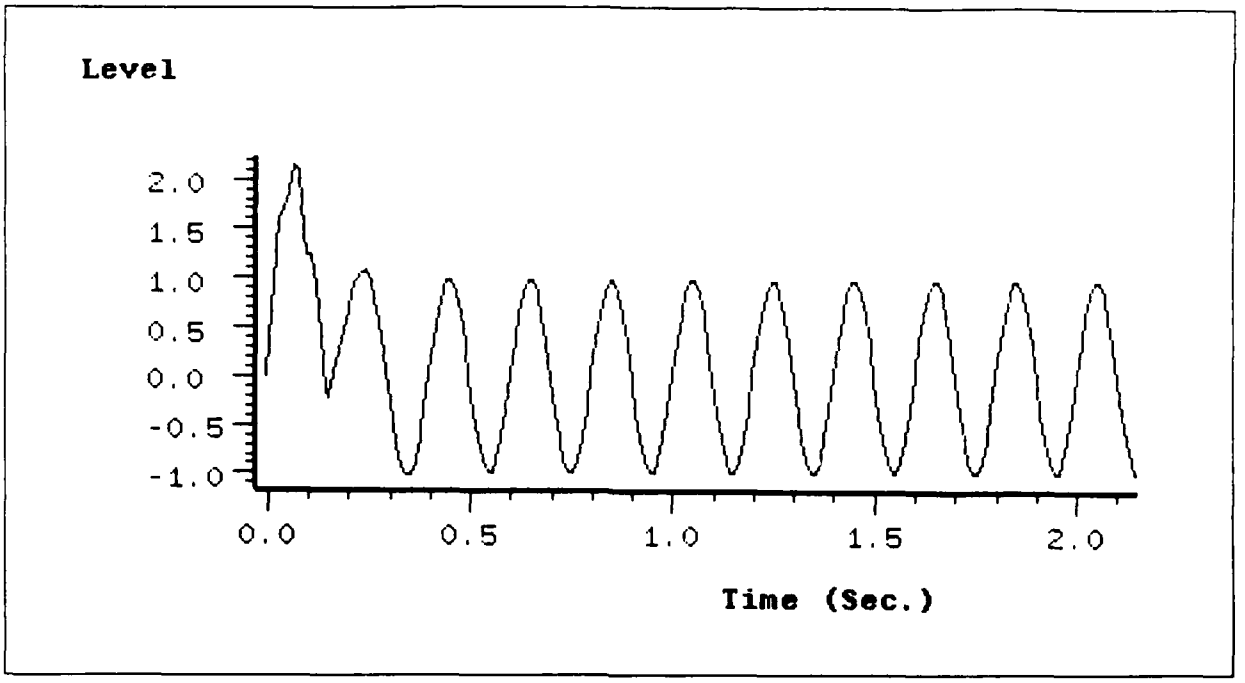

Figure 66. Four Element Array Output with Source at 0 Degrees

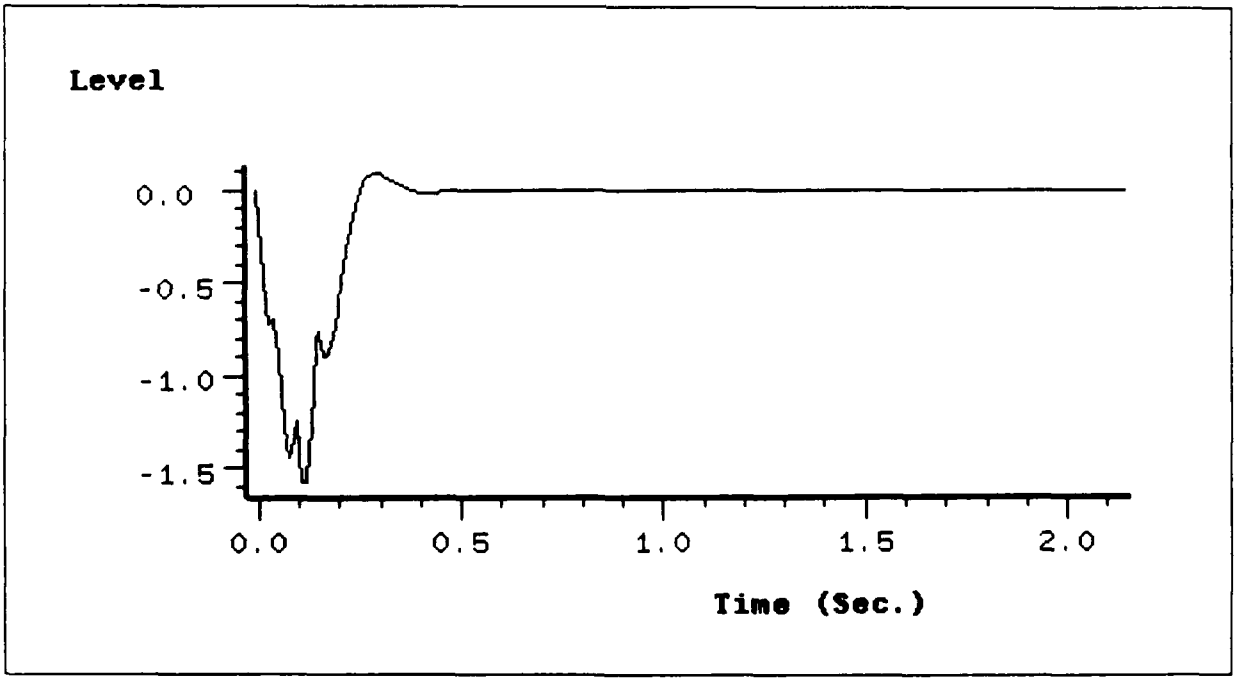

Figure 67. Four Element Array Error with Source at 0 Degrees

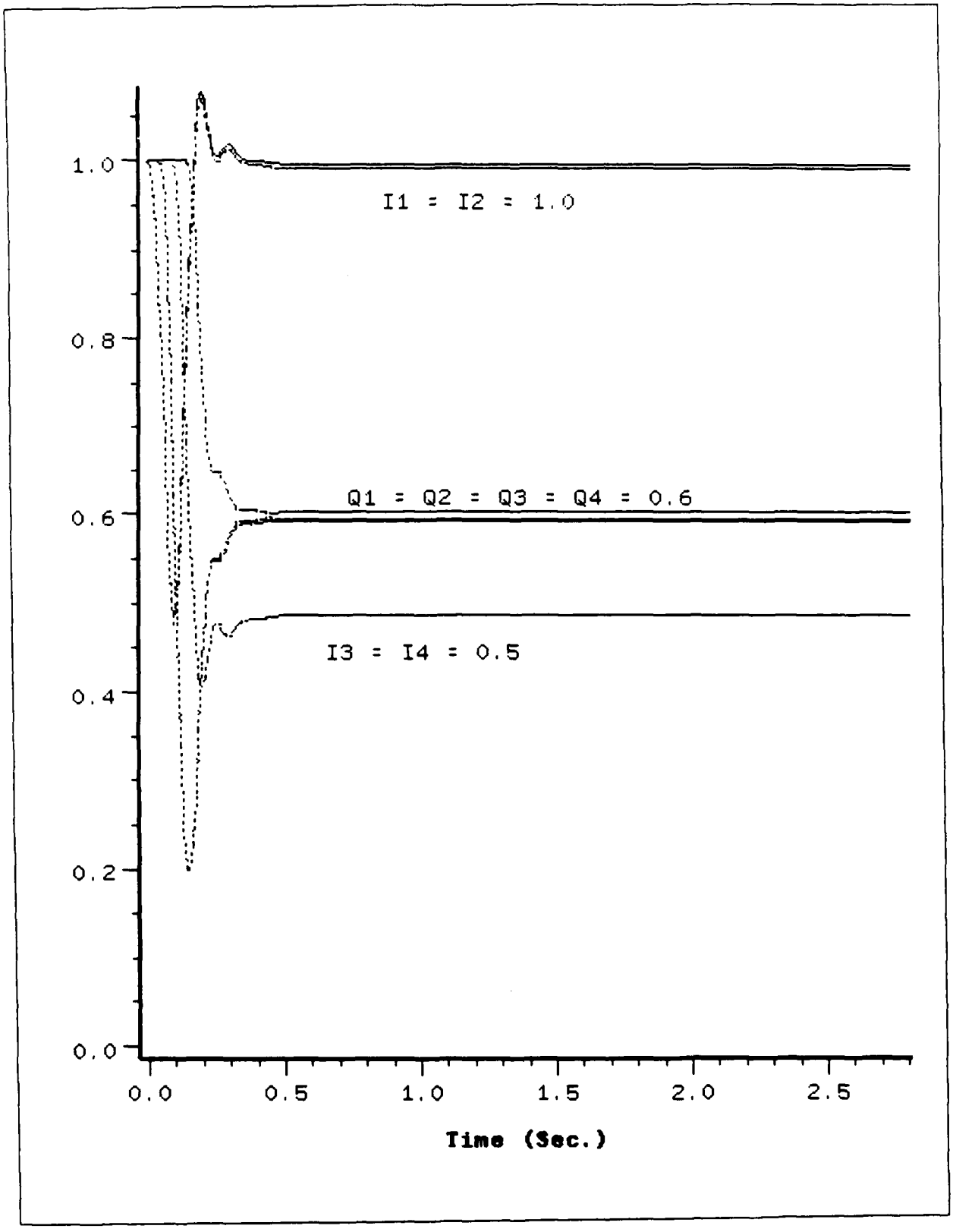

Figure 68. Four Element Array Weights with Source at 0 Degrees

```
Menu   FOUR LMS LOOPS-TEST 1 W/SOURCE AT 30 DEGREES
STOP-TIME = 10.0
DT = 5.555556E-4
DESIRED FREQ (HZ) = 5.0
DESIRED PHASE (RADS) = 0.0
DESIRED AMPLITUDE = 1.0
1/4FO = 90
CONVERGENCE FACTOR  = 5.0E-3
SIGNAL DELAY FOR ELEMENT 2 = 270
SIGNAL DELAY FOR ELEMENT 3 = 156
SIGNAL DELAY FOR ELEMENT 4 = 66
JAMMER DELAY FOR ELEMENT 4 = 66
JAMMER DELAY FOR ELEMENT 3 = 156
JAMMER DELAY FOR ELEMENT 2 = 270
NOISE DELAY FOR ELEMENT 4 = 180
NOISE DELAY FOR ELEMENT 3 = 180
NOISE DELAY FOR ELEMENT 2 = 1
NOISE POWER = 1.0e-14
JAMMER FREQ (HZ) = 5.0
JAMMER AMPLITUDE = 1.0e-14
INPUT SIGNAL FREQ (HZ) = 5.0
INPUT SIGNAL AMPLITUDE = 1.0
```

Table 11. Parameters for Four Element Array with Source at 30 Degrees

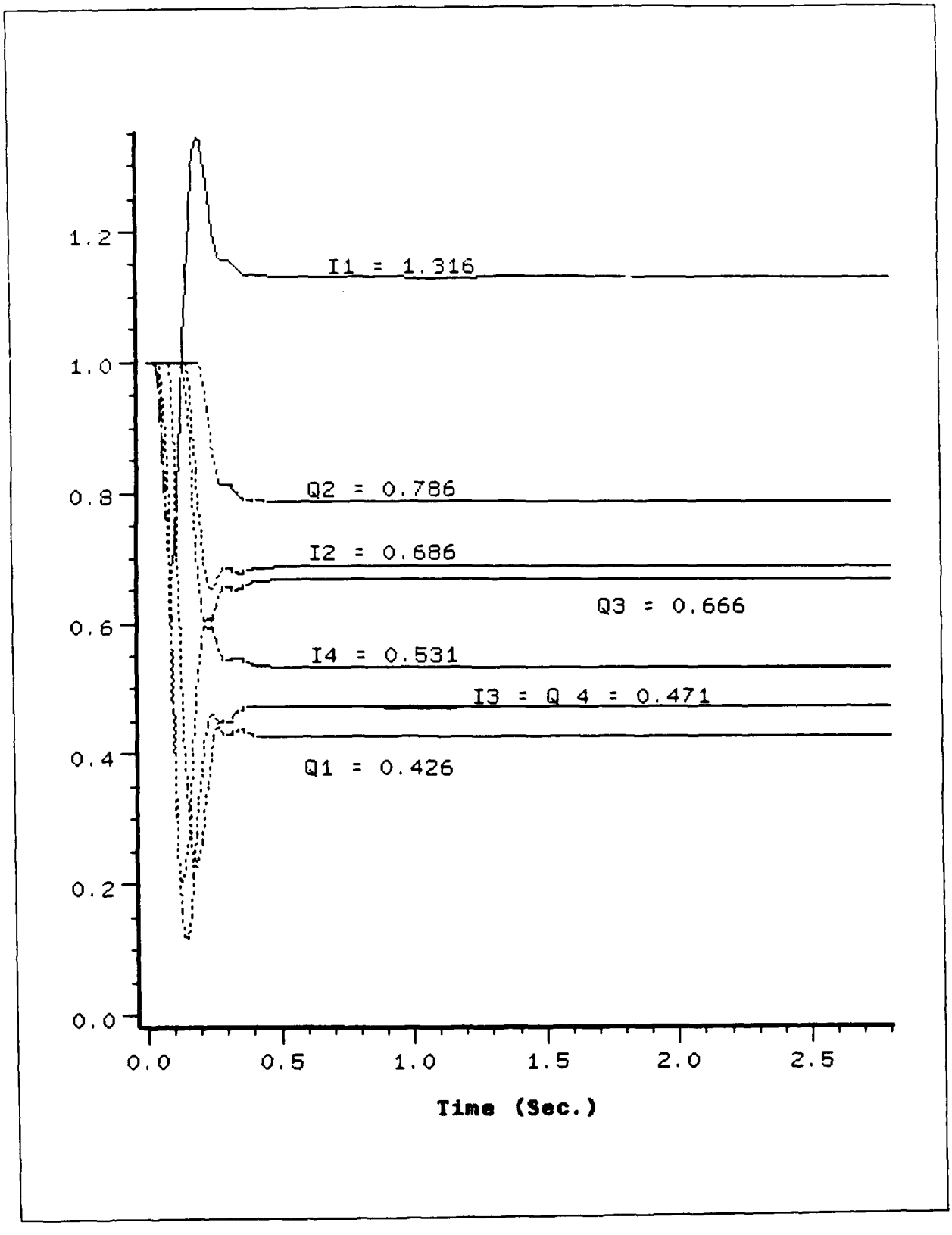

Figure 69. Four Element Array Weights with Source at 30 Degrees

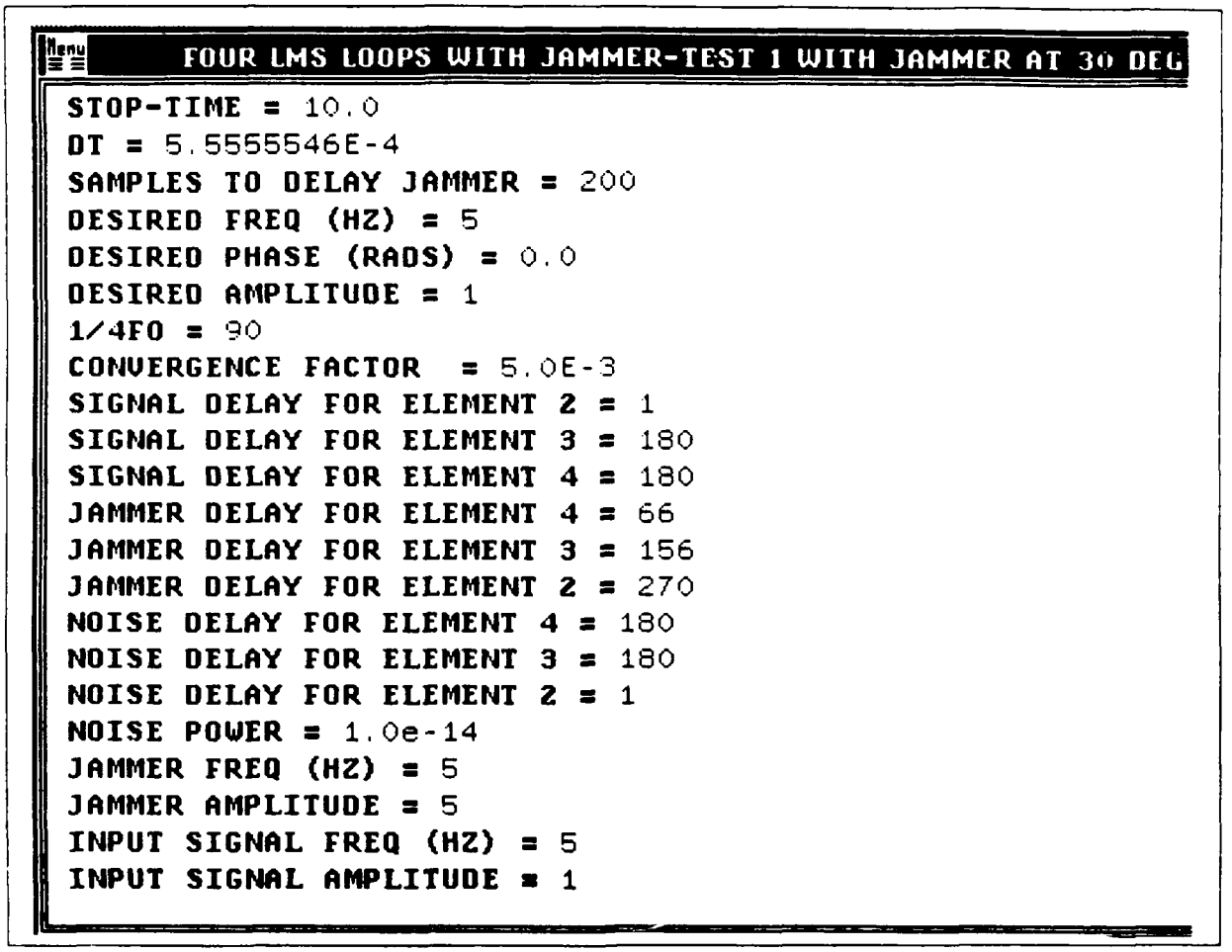

Table 12. Parameters for Four Element Array with Source at 0 Degrees and Jammer at 30 degrees

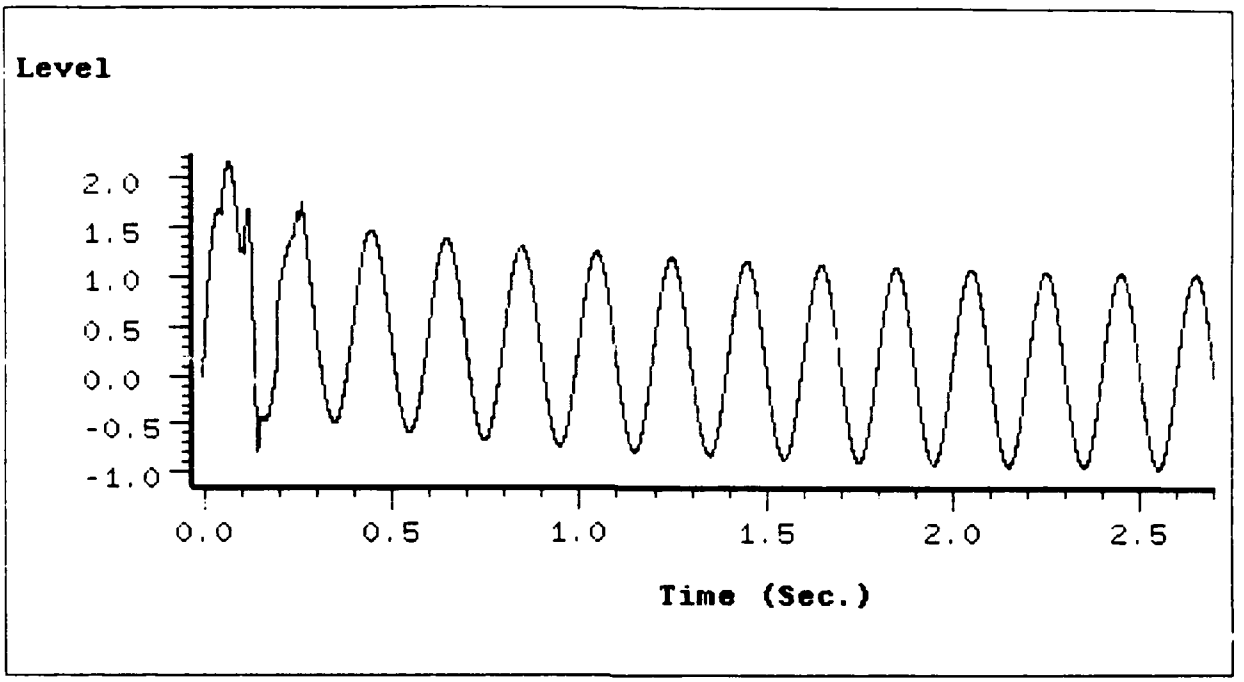

Figure 70. Four Element Array Output with Source at 0 Degrees and Jammer at 30 Degrees

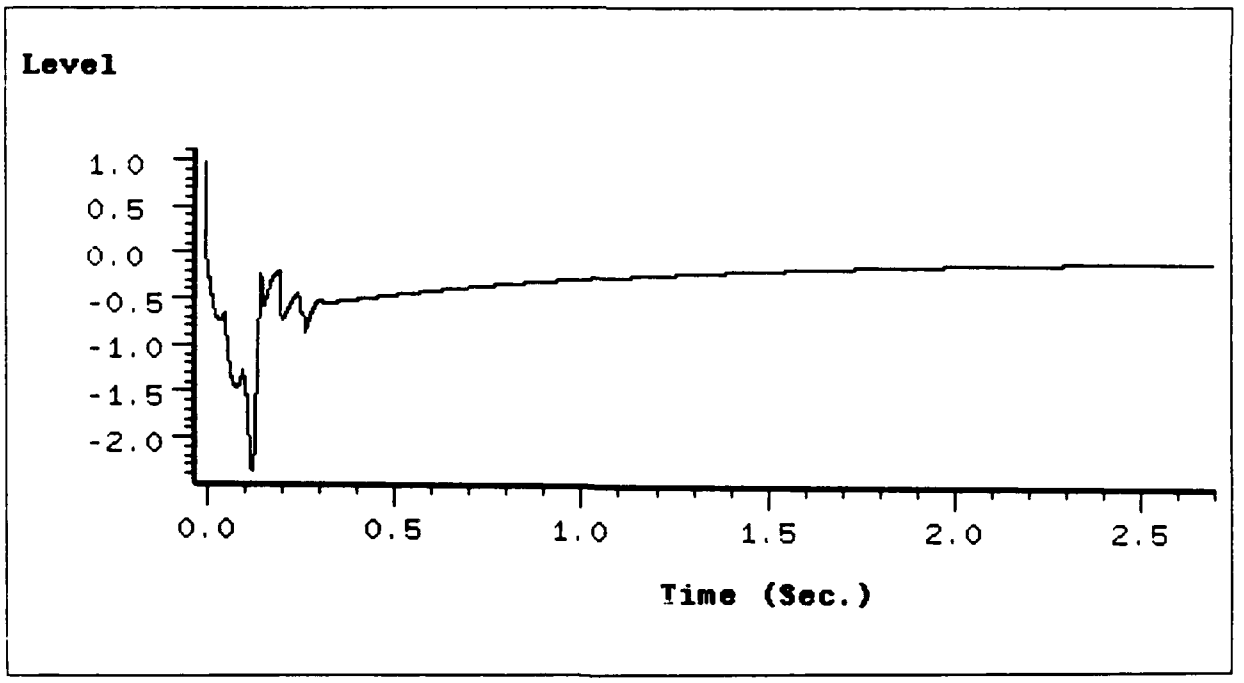

Figure 71. Four Element Array Error with Source at 0 Degrees and Jammer at 30 Degrees

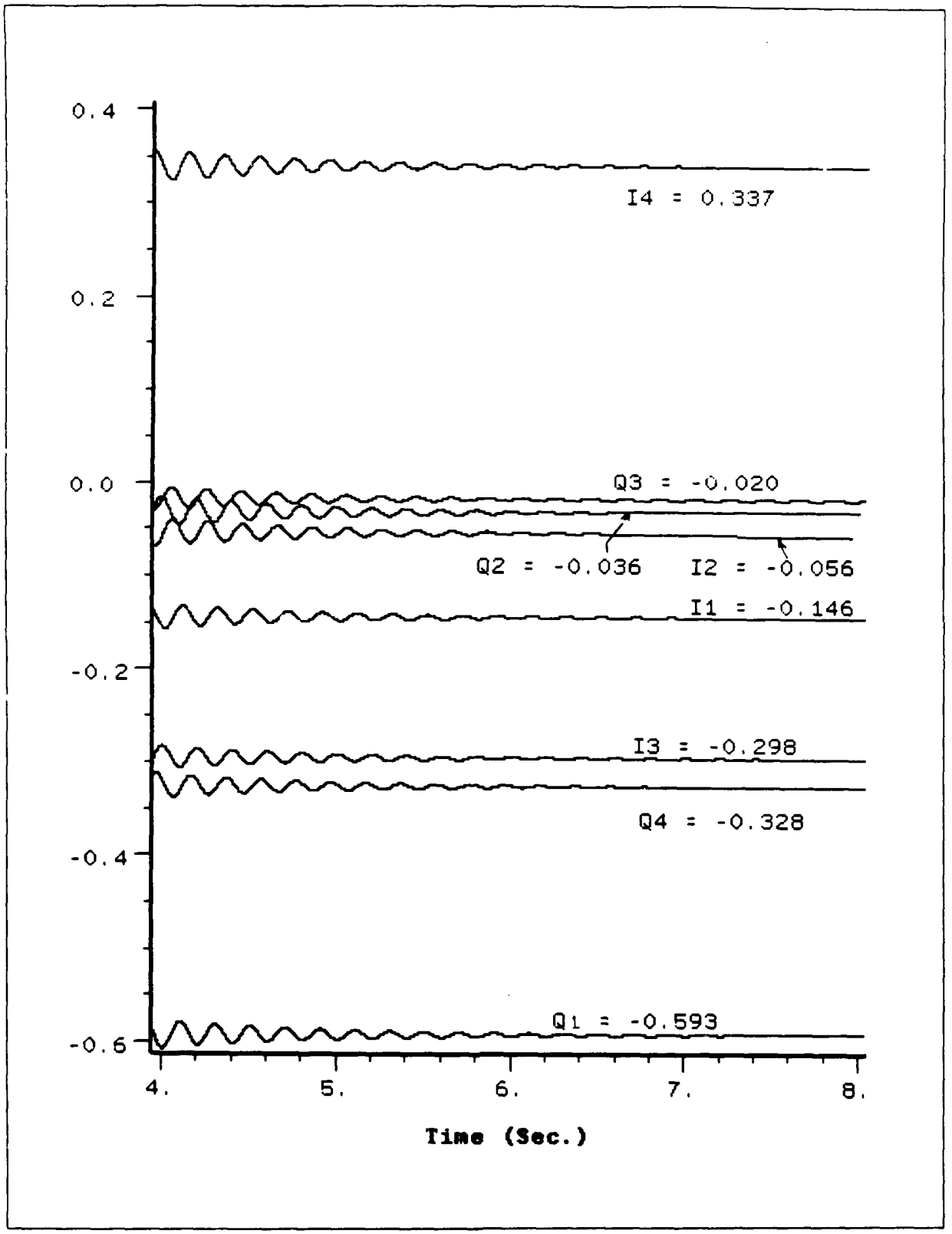

Figure 72. Four Element Array Weights with Source at 0 Degrees and Jammer at 30 Degrees

A.4 Complex LMS Test Circuit Simulation Results

Using the test system in Figure (33), simulations were performed first no phase shift by a real gain of 0.5 of the input to form the desired signal. The second test involved advancing the phase of the input signal by 30 degrees by suppling the constant generator a value of 0.5236 radians. The following two tables and seven plots show the results of the real and imaginary parts of the input, the magnitude of the error, and the real and imaginary compenents of the weights.

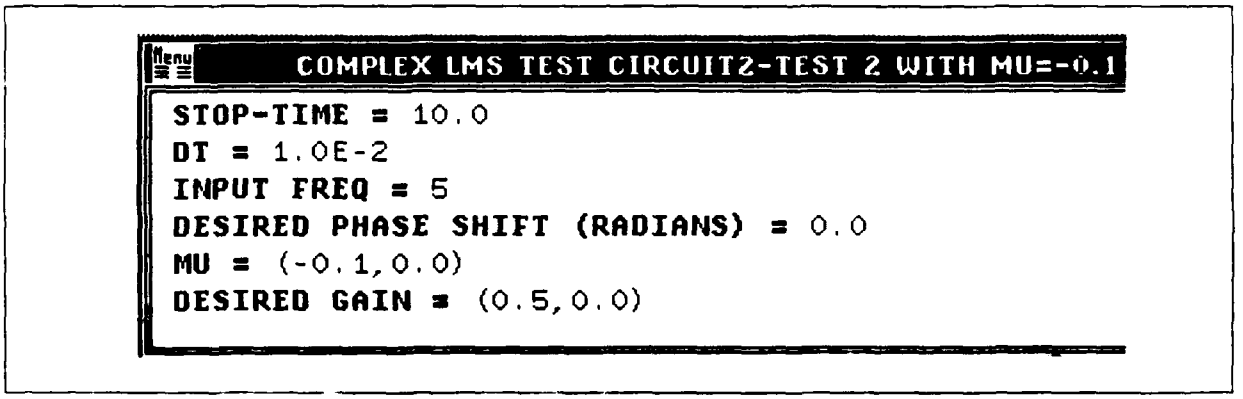

Table 13. Complex Test Circuit Parameters with 0 Degrees Phase

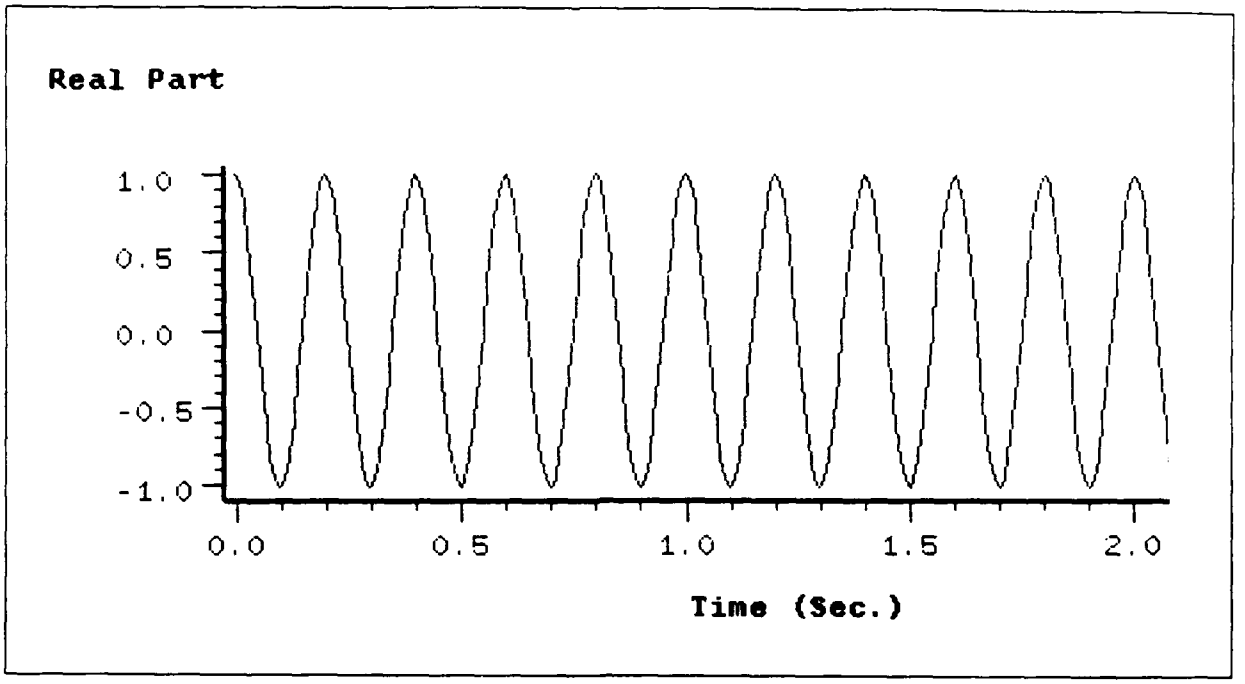

Figure 73. Real Input Component with 0 Degrees Phase

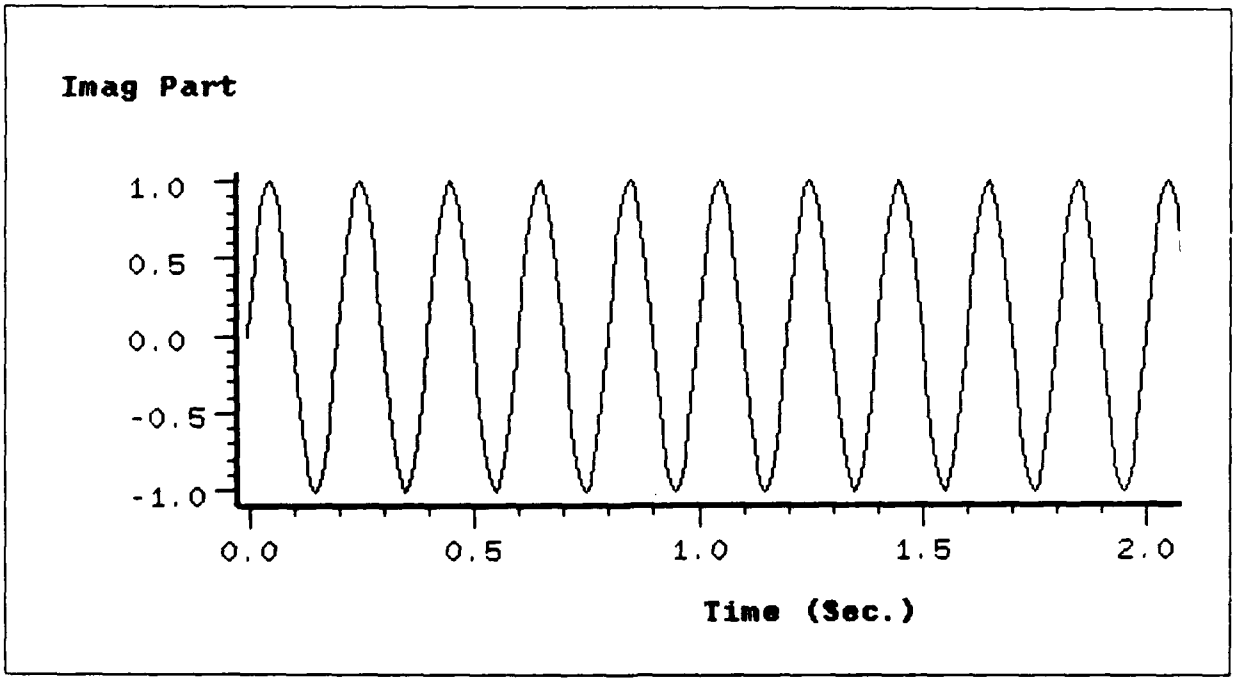

Figure 74. Imaginary Input Component with 0 Degrees Phase

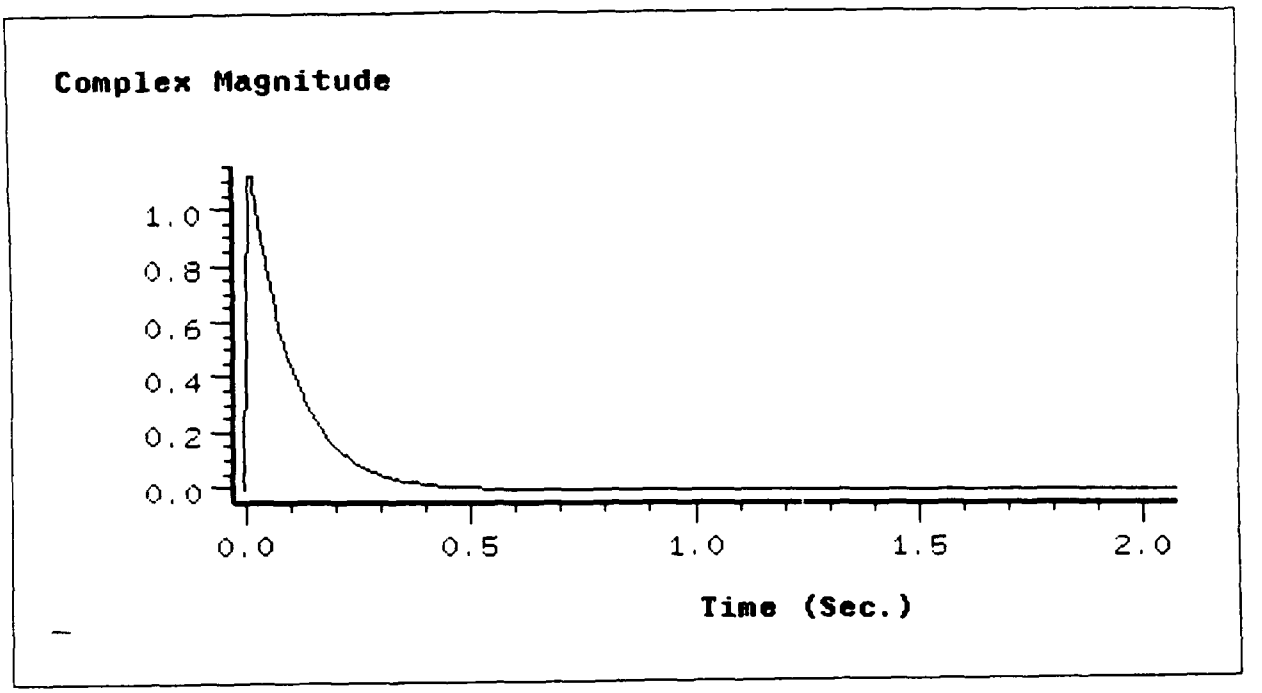

Figure 75. Error Magnitude with 0 Degrees Phase

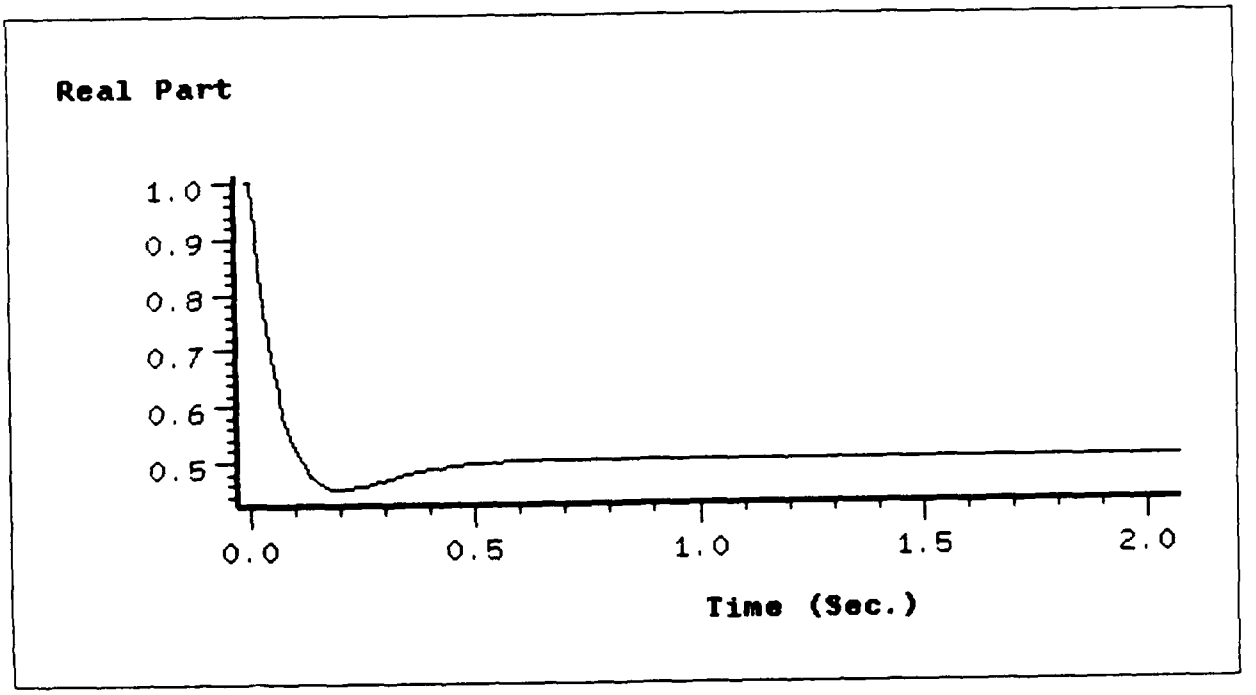

Figure 76. Real Weight Component with 0 Degrees Phase

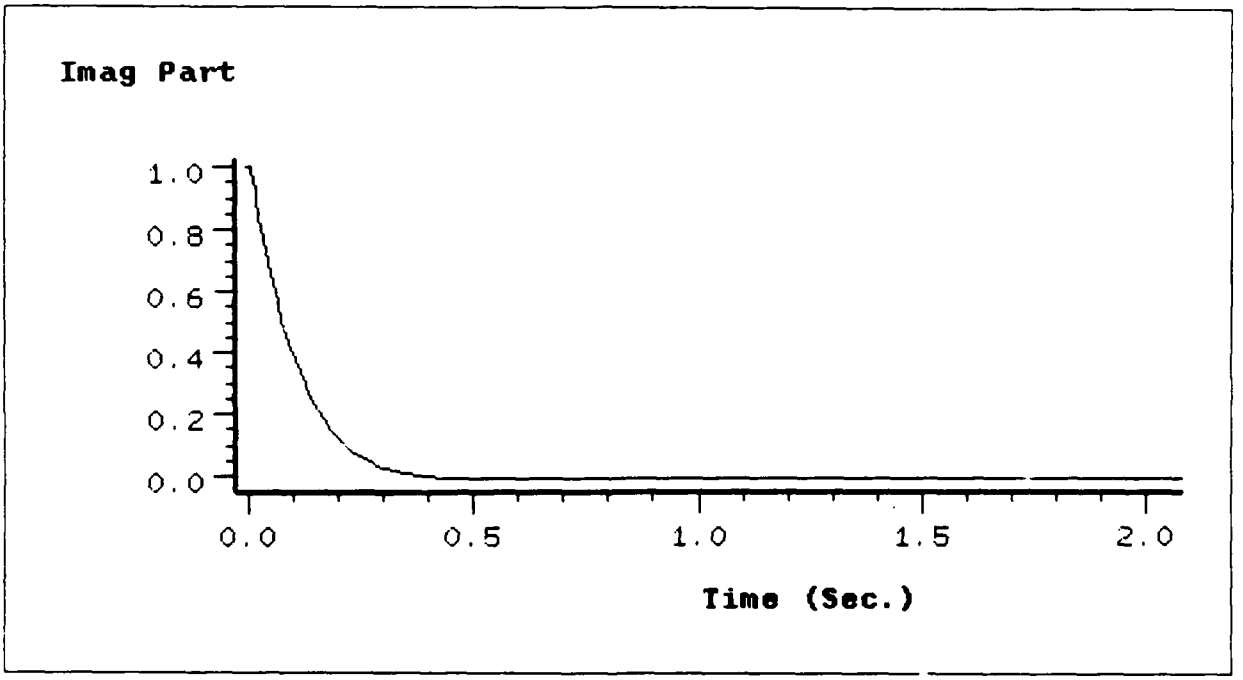

Figure 77. Imaginary Weight Component with 0 Degrees Phase

Menu COMPLEX LMS TEST CIRCUIT2-TEST 1 W/MU=-0.1, 30 DEG PHAS

```
STOP-TIME = 10.0
DT = 1.0E-2
INPUT FREQ = 5
DESIRED PHASE SHIFT (RADIANS) = 0.5236
MU = (-0.1,0.0)
DESIRED GAIN = (0.5,0.0)
```

Table 14. Complex Test Circuit Parameters with 30 Degrees Phase

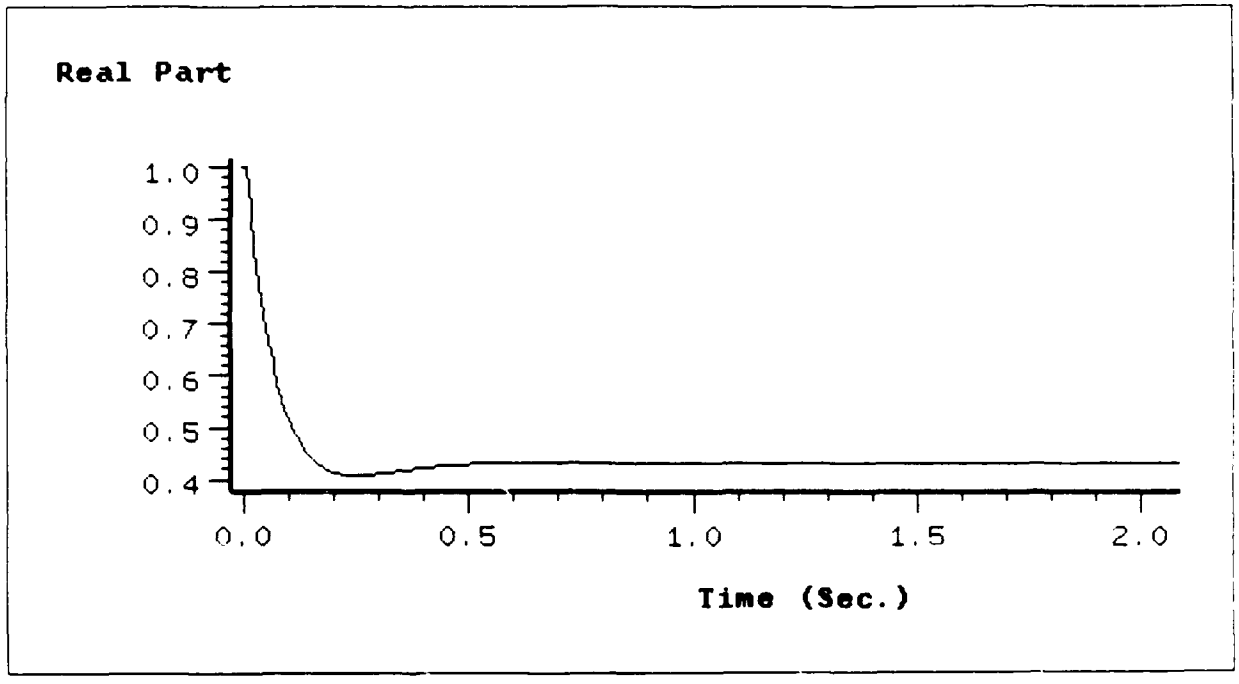

Figure 78. Real Weight Component with 30 Degrees Phase

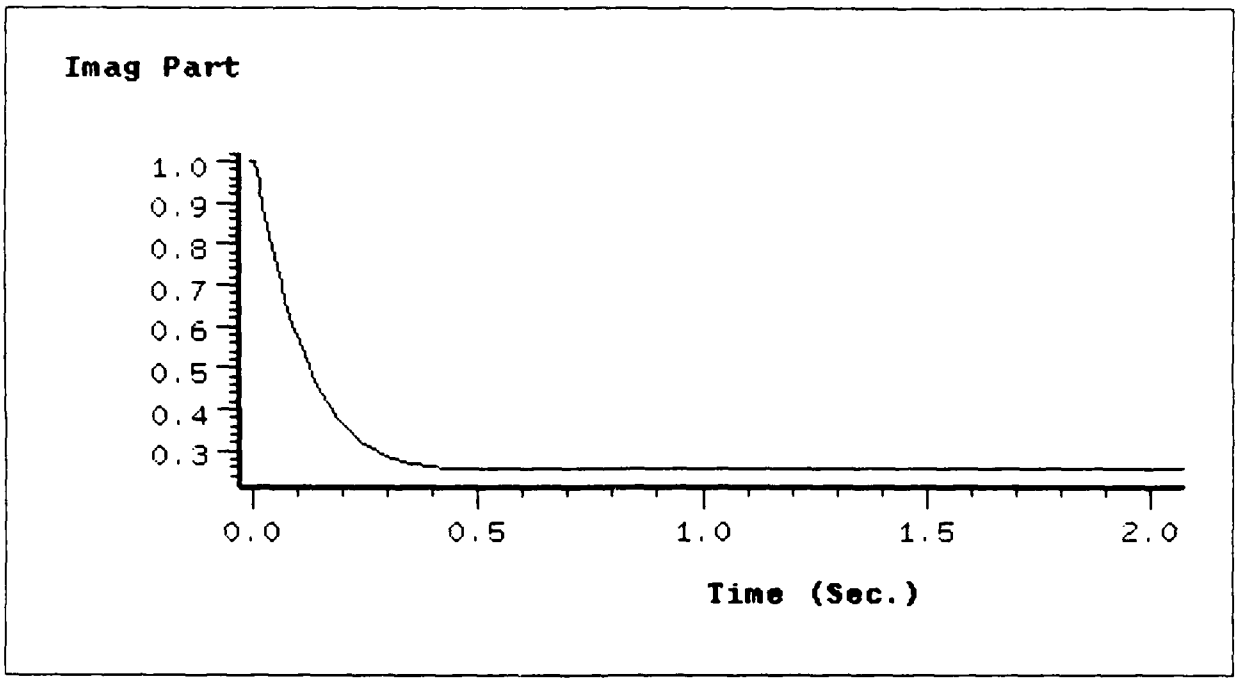

Figure 79. Imaginary Weight Component with 30 Degrees Phase

A.5 *Complex Four Element Array Simulation Results*

The first simulation of this array involved putting the source at 0 degrees with reference to the upper left array element in the square array. The jammer and noise power were virturally 0. Table (15) shows the parameters used for this simulation. The next three figures show the magitude of the complex error and output along with the real and imaginary components of the complex weights.

```
Menu   COMPLEX 4 ELEMENT SYSTEM-TEST 1 W/SOURCE AT 0 DEGREES
STOP-TIME = 10.0
DT = 1.0E-2
DESIRED FREQUENCY = 5.0
MU = (-0.01,0.0)
NOISE ELEMENT 4 PHASE (RADIANS) = -3.1415
NOISE ELEMENT 3 PHASE (RADIANS) = -3.1415
NOISE ELEMENT 2 PHASE (RADIANS) = 0.0
JAMMER ELEMENT 4 PHASE (RADIANS) = -3.1415
JAMMER ELEMENT 3 PHASE (RADIANS) = -3.1415
JAMMER ELEMENT 2 PHASE (RADIANS) = 0.0
SIGNAL ELEMENT 4 PHASE (RADIANS) = -3.1415
SIGNAL ELEMENT 3 PHASE (RADIANS) = -3.1415
SIGNAL ELEMENT 2 PHASE (RADIANS) = 0.0
NOISE POWER = 3.0e-14
JAMMER AMPLITUDE = (1.0e-14,0.0)
SIGNAL AMPLITUDE = (1.0,0.0)
JAMMER FREQUENCY = 5.0
SIGNAL FREQUENCY = 5.0
```

Table 15. Complex Four Element Array Parameters with Source at 0 Degrees

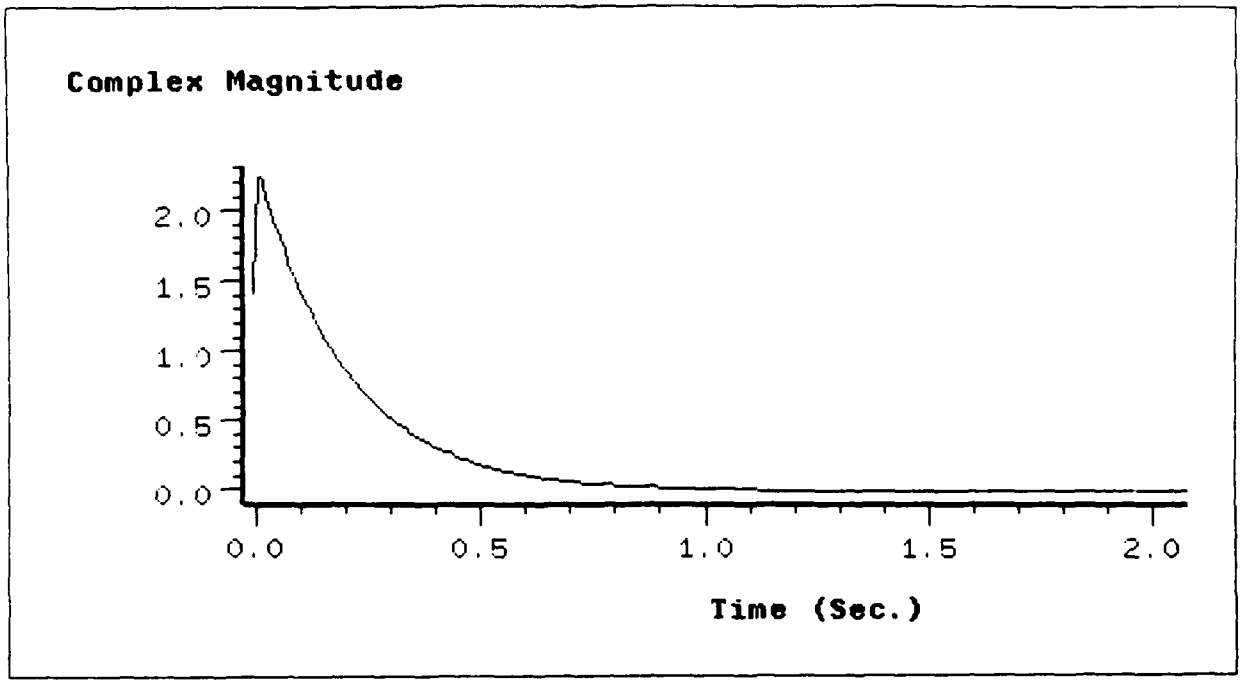

Figure 80. Complex Four Element Array Error Magnitude with Source at 0 Degrees

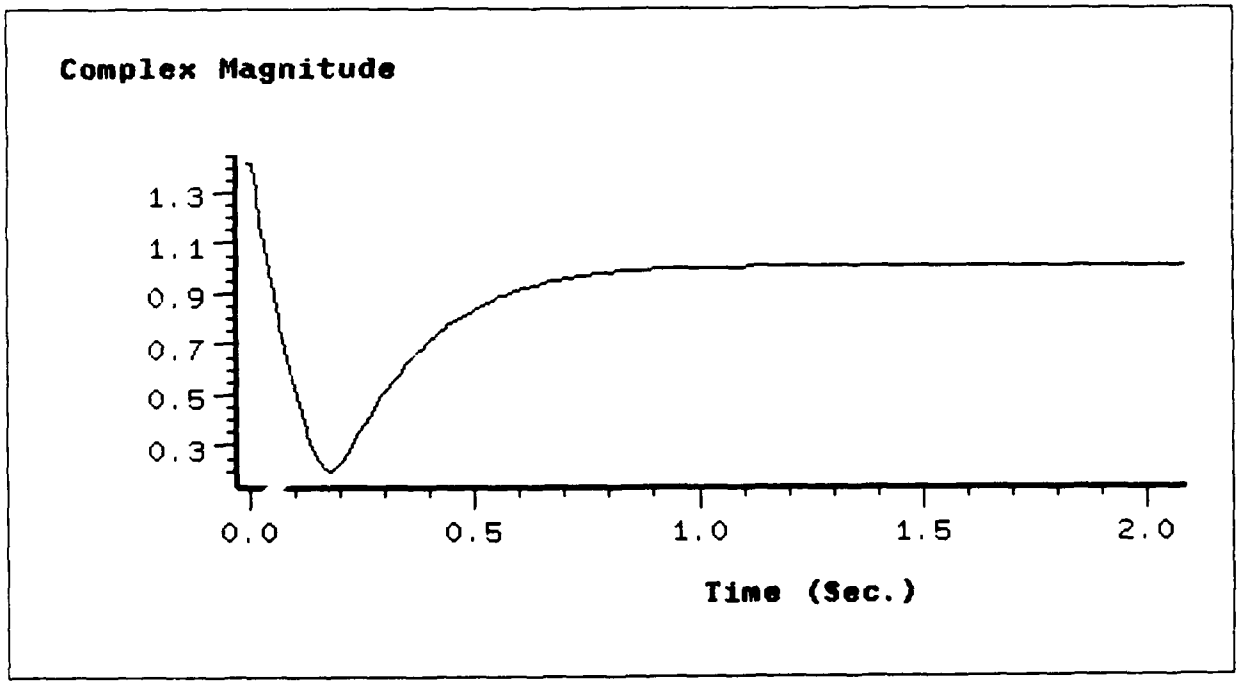

Figure 81. Complex Four Element Array Output Magnitude with Source at 0 Degrees

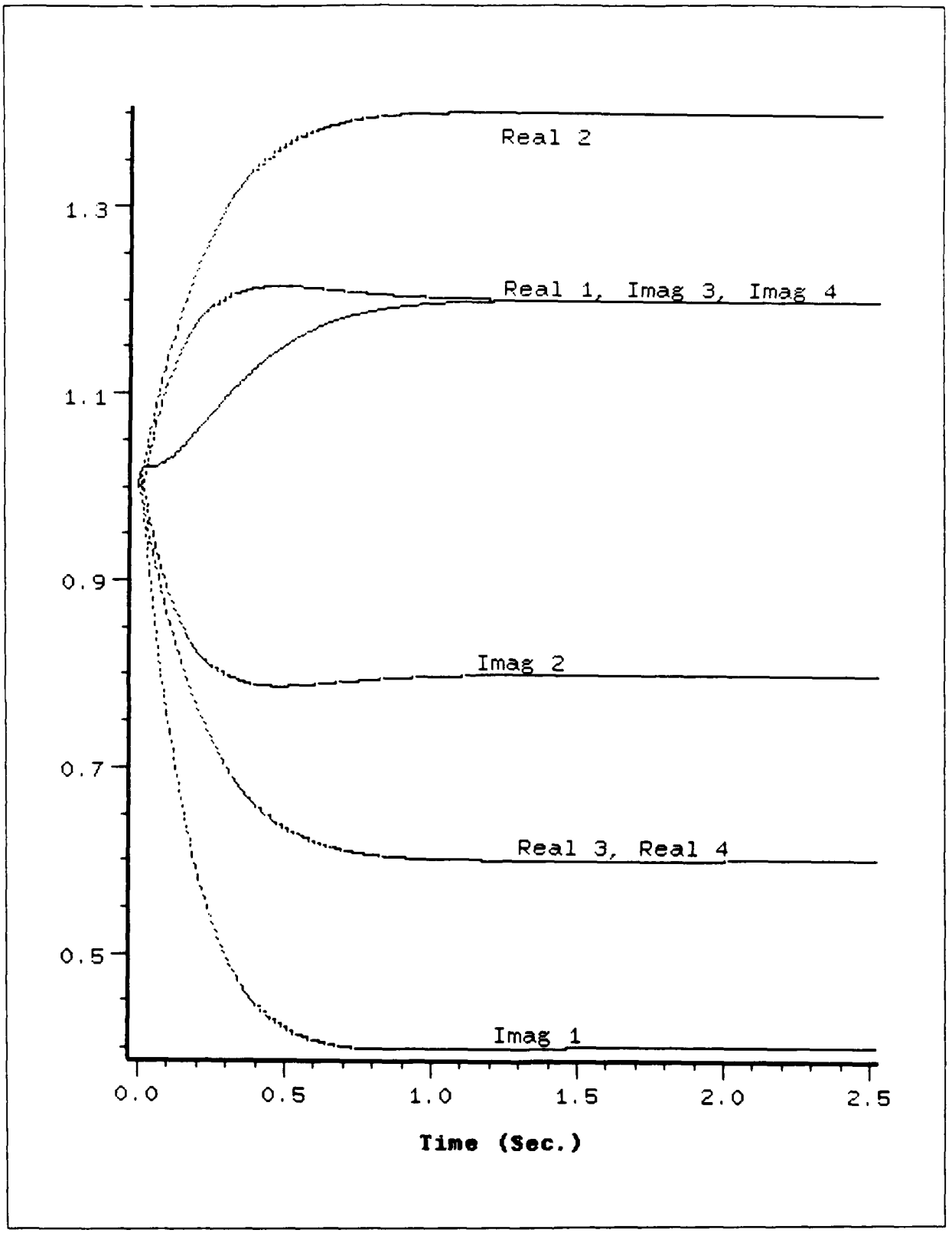

Figure 82. Complex Four Element Array Weights with Source at 0 Degrees

The source was then moved to 30 degrees. The sources magnitude was kept at 1 with the a frequency of 5 Hz. The simulation parameters are in Table (16) with the error, output, and weight plots in Figures (83), (84), and (85).

```
Menu   COMPLEX 4 ELEMENT SYSTEM-TEST 1 W/ JAMMER AT 30 DEGREES
STOP-TIME = 10.0
DT = 1.0E-2
DESIRED FREQUENCY = 5.0
MU = (-0.001,0.0)
NOISE ELEMENT 4 PHASE (RADIANS) = -3.1415
NOISE ELEMENT 3 PHASE (RADIANS) = -3.1415
NOISE ELEMENT 2 PHASE (RADIANS) = 0.0
JAMMER ELEMENT 4 PHASE (RADIANS) = -1.1499
JAMMER ELEMENT 3 PHASE (RADIANS) = -2.72069
JAMMER ELEMENT 2 PHASE (RADIANS) = 1.570796
SIGNAL ELEMENT 4 PHASE (RADIANS) = -3.1415
SIGNAL ELEMENT 3 PHASE (RADIANS) = -3.1415
SIGNAL ELEMENT 2 PHASE (RADIANS) = 0.0
NOISE POWER = 1.0e-14
JAMMER AMPLITUDE = (5.0,0.0)
SIGNAL AMPLITUDE = (1.0,0.0)
JAMMER FREQUENCY = 5.0
SIGNAL FREQUENCY = 5.0
```

Table 16. Complex Four Element Array Parameters with Source at 30 Degrees

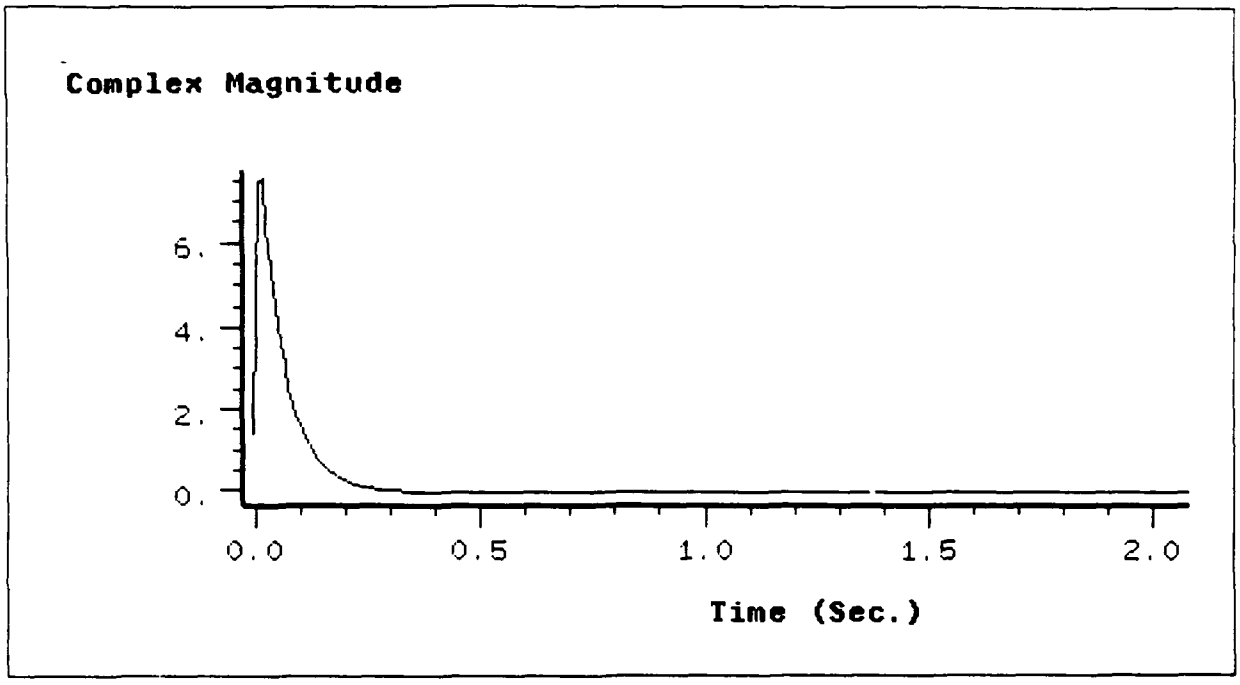

Figure 83. Complex Four Element Array Error Magnitude with Source at 30 Degrees

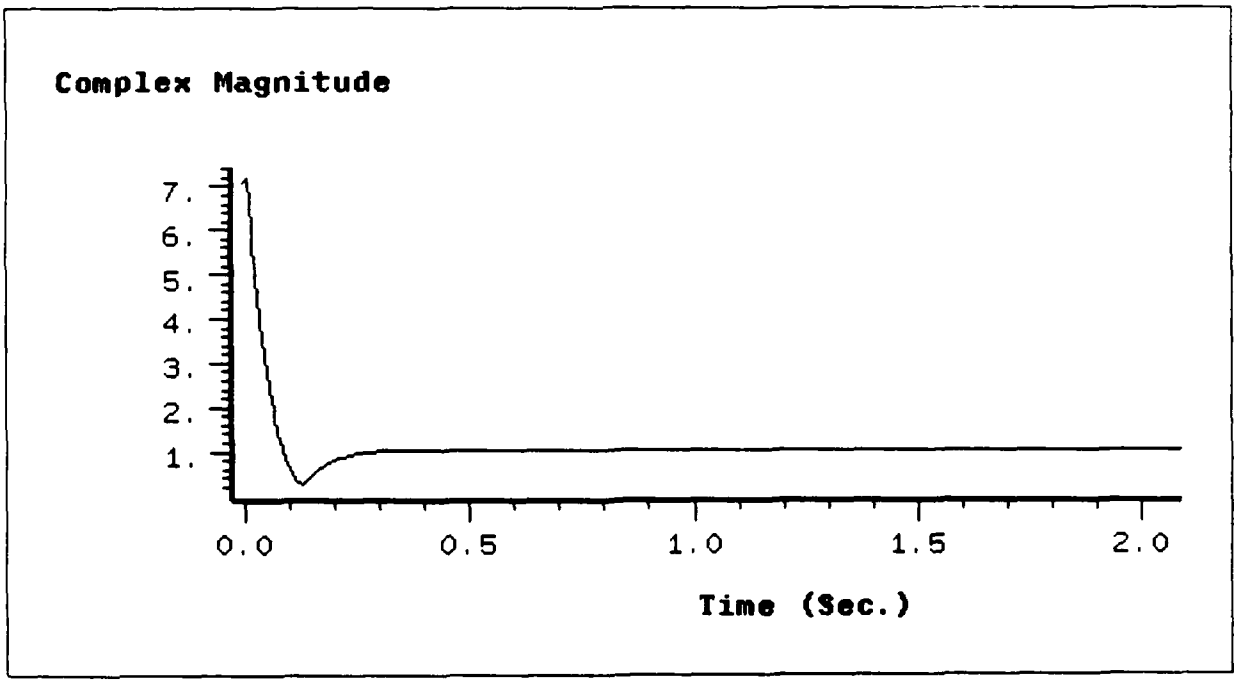

Figure 84. Complex Four Element Array Output Magnitude with Source at 30 Degrees

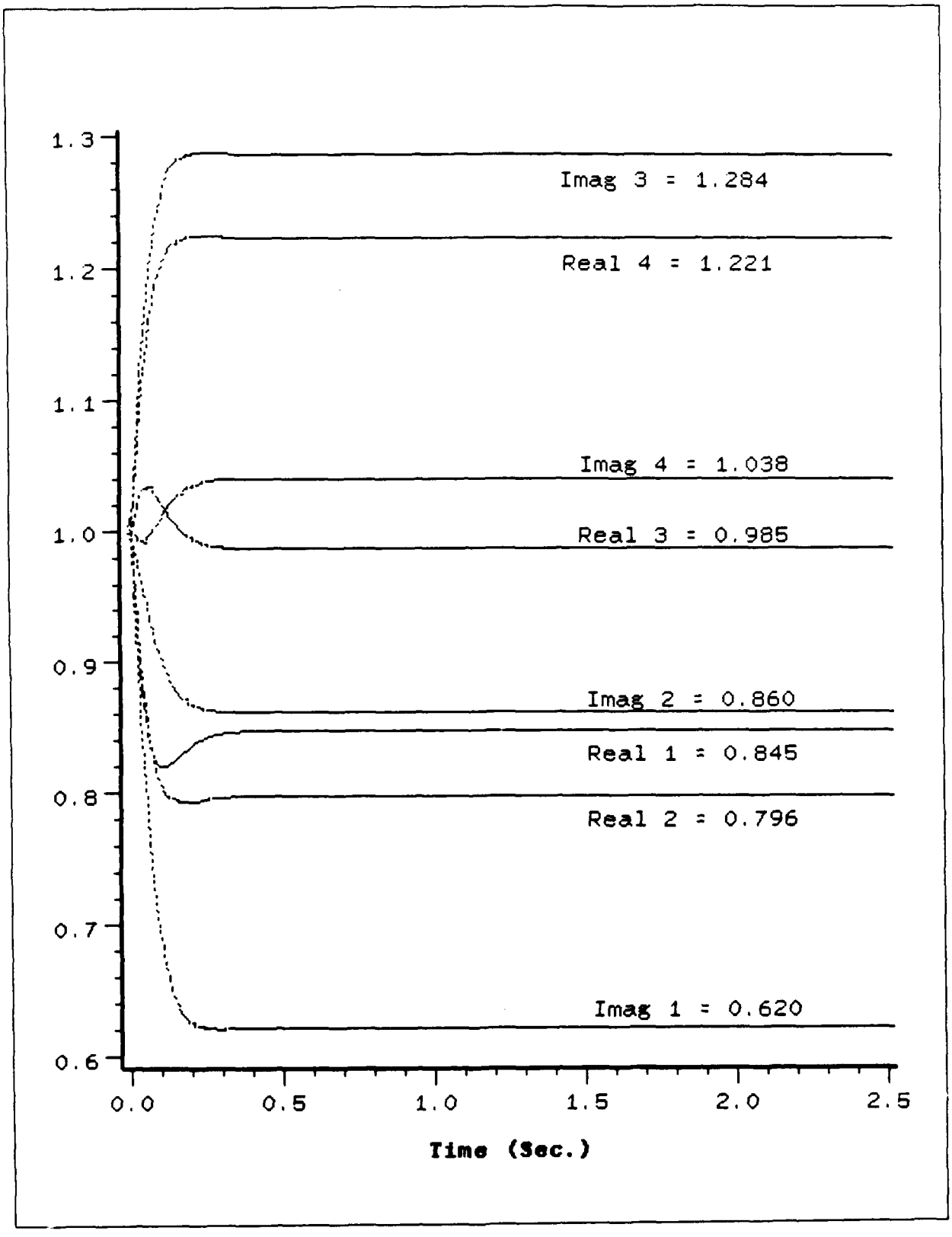

Figure 85. Complex Four Element Array Weights with Source at 30 Degrees

The last simulation for this system moved the source back to 0 degrees and placed a jammer at 30 degrees. The jammer and source placements were accomplished by changing the phase on the appropriate signal and jammer elements. The jammer amplitude was set to 5 while the signal and desired amplitudes were set to 1. The noise power was set to 0. Each of these parameters are contained in Table (17). The following three plots contain the error and output magnitudes and the real and imaginary weight components.

COMPLEX 4 ELEMENT SYSTEM-TEST 1 W/SOURCE AT 30 DEGREES
STOP-TIME = 10.0
DT = 1.0E-2
DESIRED FREQUENCY = 5.0
MU = (-0.01,0.0)
NOISE ELEMENT 4 PHASE (RADIANS) = -3.1415
NOISE ELEMENT 3 PHASE (RADIANS) = -3.1415
NOISE ELEMENT 2 PHASE (RADIANS) = 0.0
JAMMER ELEMENT 4 PHASE (RADIANS) = -3.1415
JAMMER ELEMENT 3 PHASE (RADIANS) = -3.1415
JAMMER ELEMENT 2 PHASE (RADIANS) = 0.0
SIGNAL ELEMENT 4 PHASE (RADIANS) = -1.1499
SIGNAL ELEMENT 3 PHASE (RADIANS) = -2.72069
SIGNAL ELEMENT 2 PHASE (RADIANS) = 1.570796
NOISE POWER = 1.0e-14
JAMMER AMPLITUDE = (1.0e-14,0.0)
SIGNAL AMPLITUDE = (1.0,0.0)
JAMMER FREQUENCY = 5.0
SIGNAL FREQUENCY = 5.0

Table 17. Complex Four Element Array Parameters with Jammer at 30 Degrees

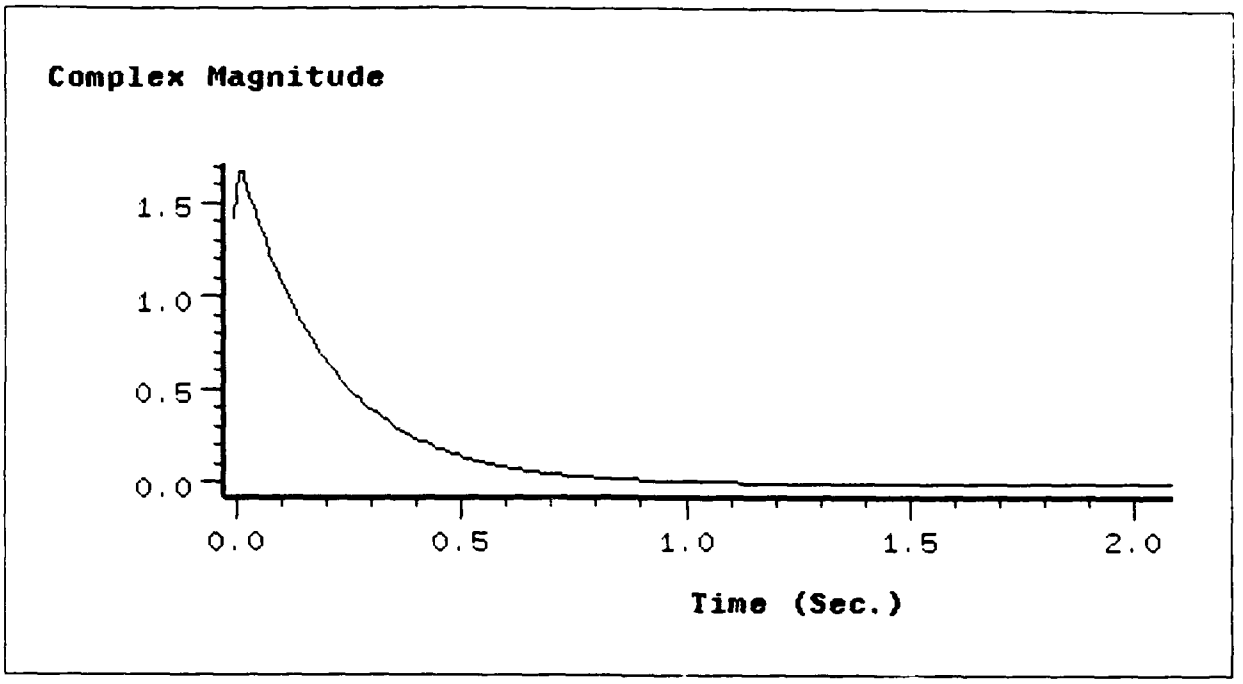

Figure 86. Complex Four Element Array Error Magnitude with Jammer at 30 Degrees

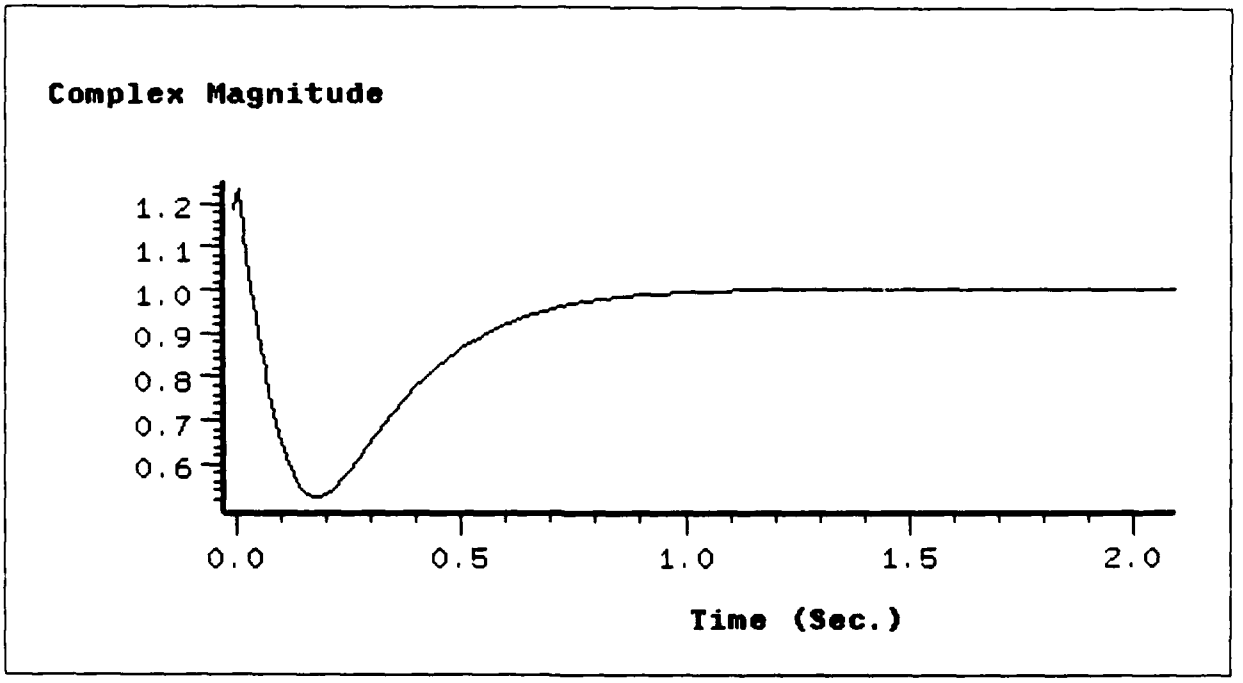

Figure 87. Complex Four Element Array Output Magnitude with Jammer at 30 Degrees

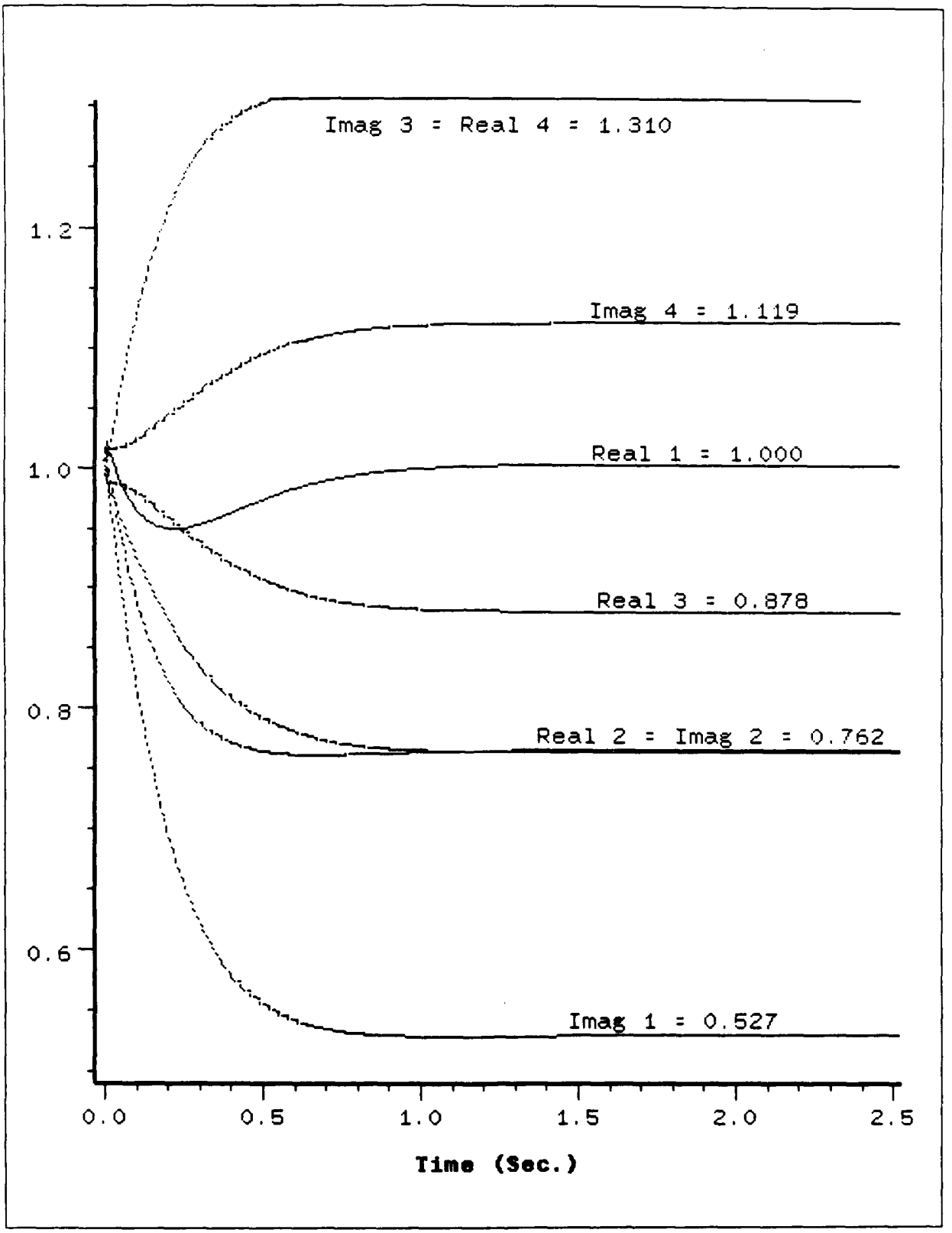

Figure 88. Complex Four Element Array Weights with Jammer at 30 Degrees

A.6 *Applebaum Sidelobe Canceller*

The Applebaum algorithm was tested using a sidelobe canceller circuit. A 20 dB jammer was placed at 10 degrees referenced to broadside of the two element array. Using the analog implementation of the Applebaum loop, Figures (89), (90) show the weight magnitude and phase during adaptation. Figure (91) shows the input spectrum with the 20 dB jammer at 50 Hz while Figure (92) shows the output signal spectrum after adaptation. Figure (93) shows the array gain as a function of spatial position with a 16 dB null at 10 degrees.

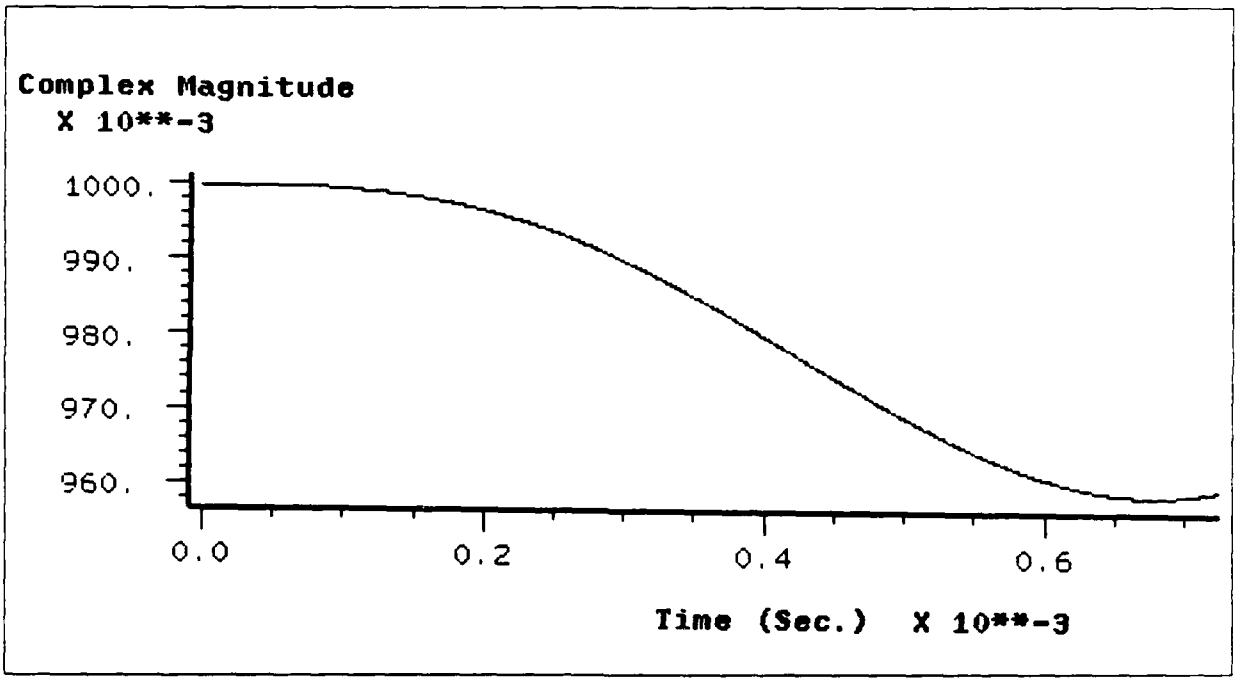

Figure 89. **Applebaum** Analog Weight Magnitude - Sidelobe Canceller

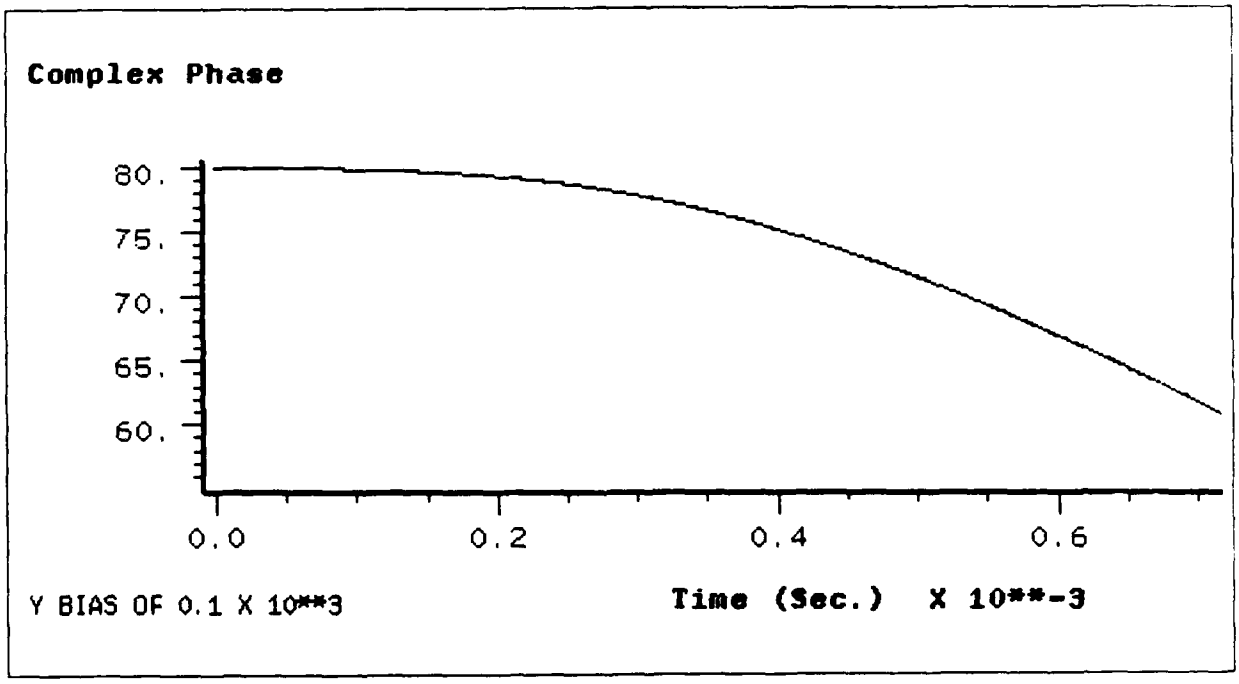

Figure 90. Applebaum Analog Weight Phase - Sidelobe Canceller

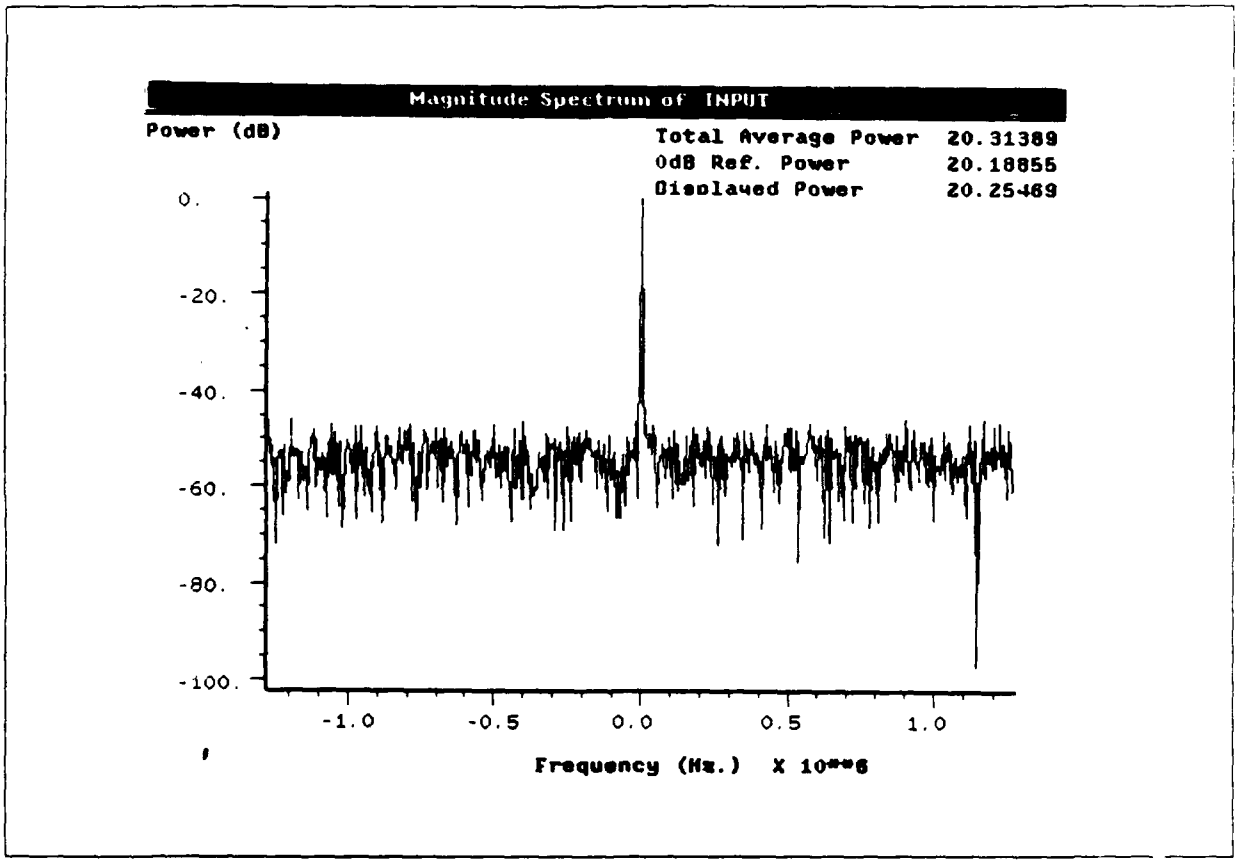

Figure 91. Input Spectrum with 20 dB Jammer at 50 Hz

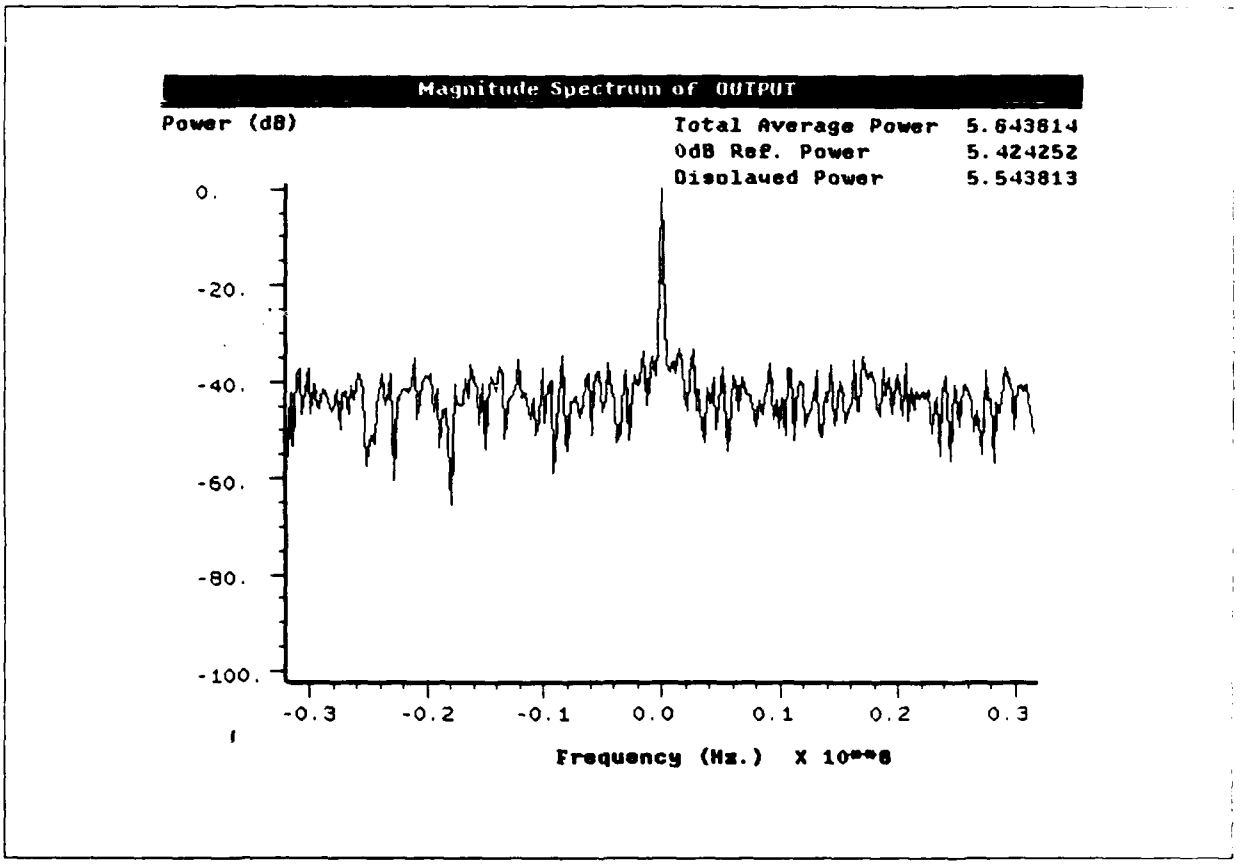

Figure 92. Output Spectrum after Adaptation with Applebaum Analog Update Algorithm

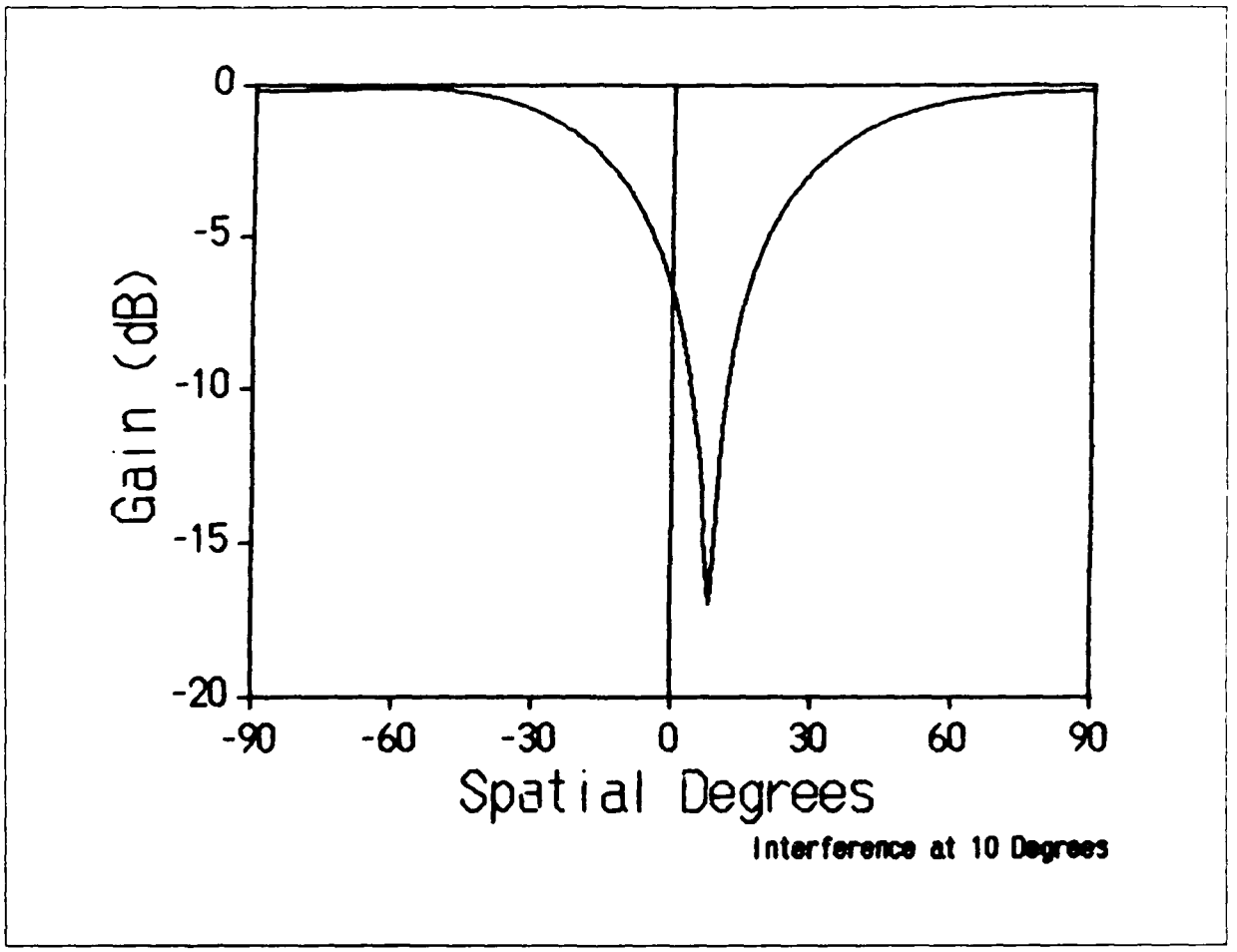

Figure 93. Array Gain with Jammer at 10 Degrees

The analog loop was then replaced with a discrete implementation of the Applebaum algorithm and the circuit was retested. Table (18) contains the parameters used for this simulation. Figures (94) and (95) show the weight magnitude and phase converging during the simulation. The next two figures show the input and output magnitude spectrums followed by the array gain plot.

```
STOP-TIME = 3.9999998E-4
DT = 5.0e-8
GAMMA = (0.01,0.0)
MU = (0.01,0.0)
LOWPASS INPUT FREQ = 5.0e6
B2 = (1.0,0.0)
MU1 = 0.0
JAMMER FREQ = 50
JAMMER NOISE POWER = 1.0E-3
JAMMER AMPLITUDE = (4.5,0.0)
PHASE SHIFT ELEMENT 1 = -0.272766
PHASE SHIFT ELEMENT 2 = 0.272766
N2 POWER = 0.1
N1 POWER = 0.1
MU0 = 0.0
```

Table 18. Discrete Applebaum Algorithm Sidelobe Canceller Parameters with Jammer at 10 Degrees

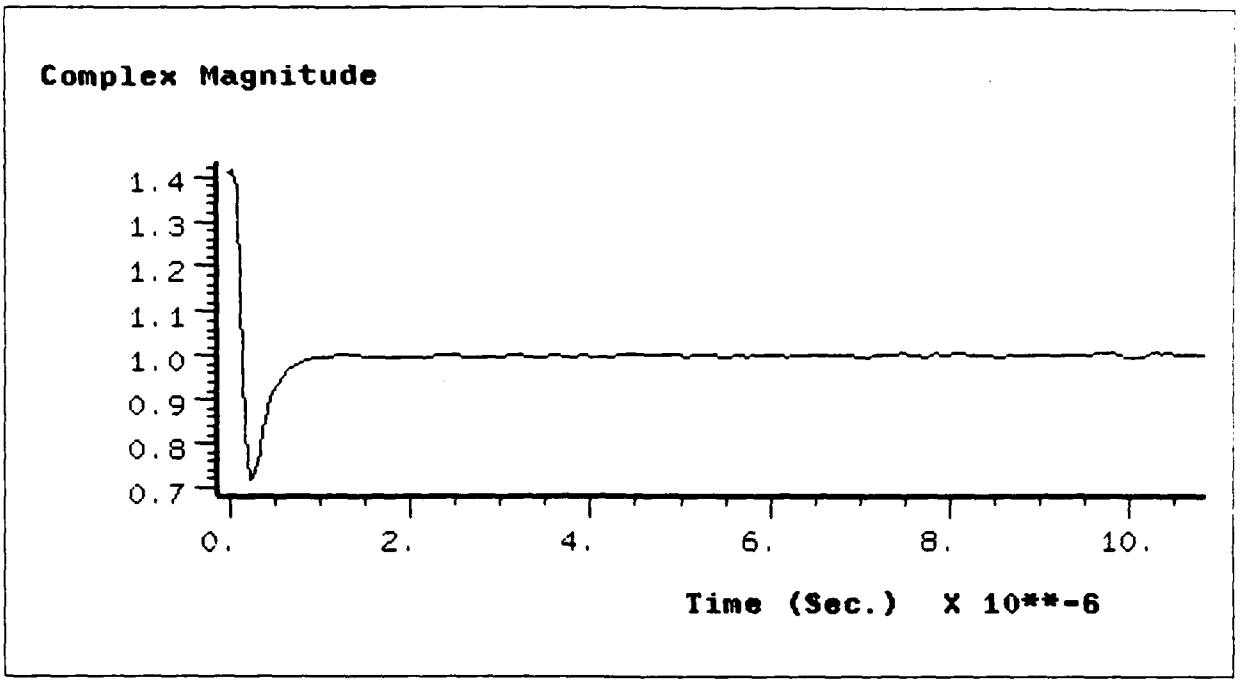

Figure 94. Applebaum Discrete Weight Magnitude - Sidelobe Canceller

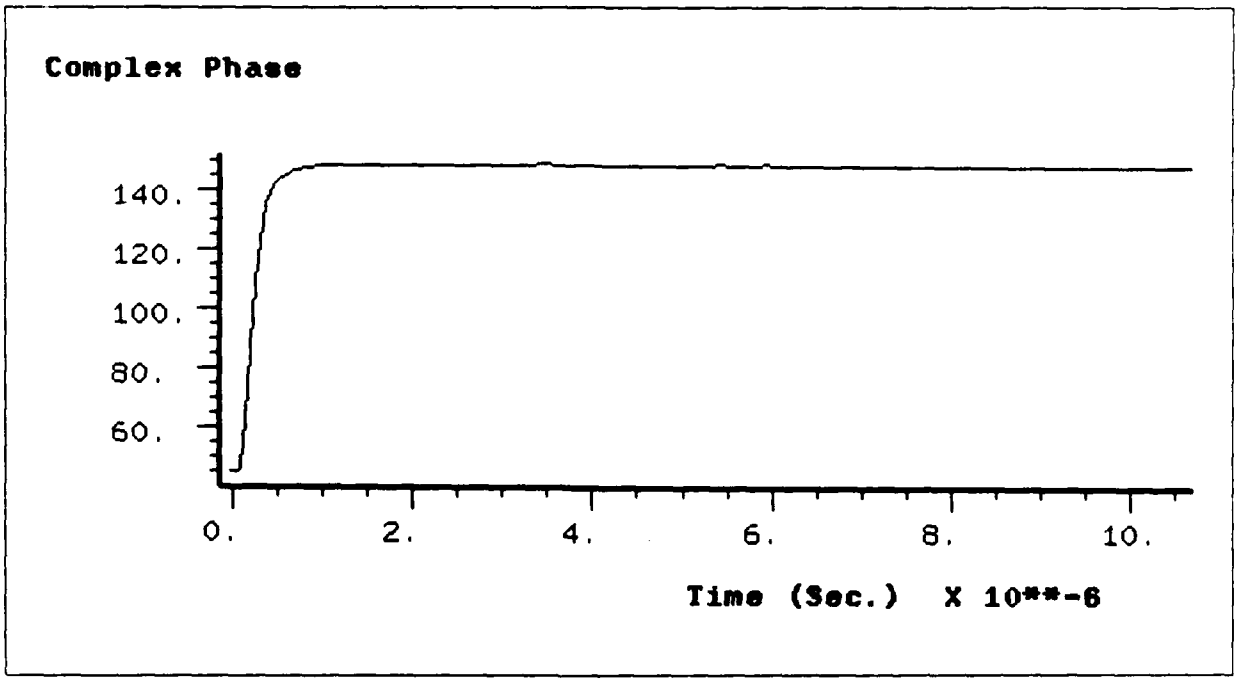

Figure 95. Applebaum Discrete Weight Phase - Sidelobe Canceller

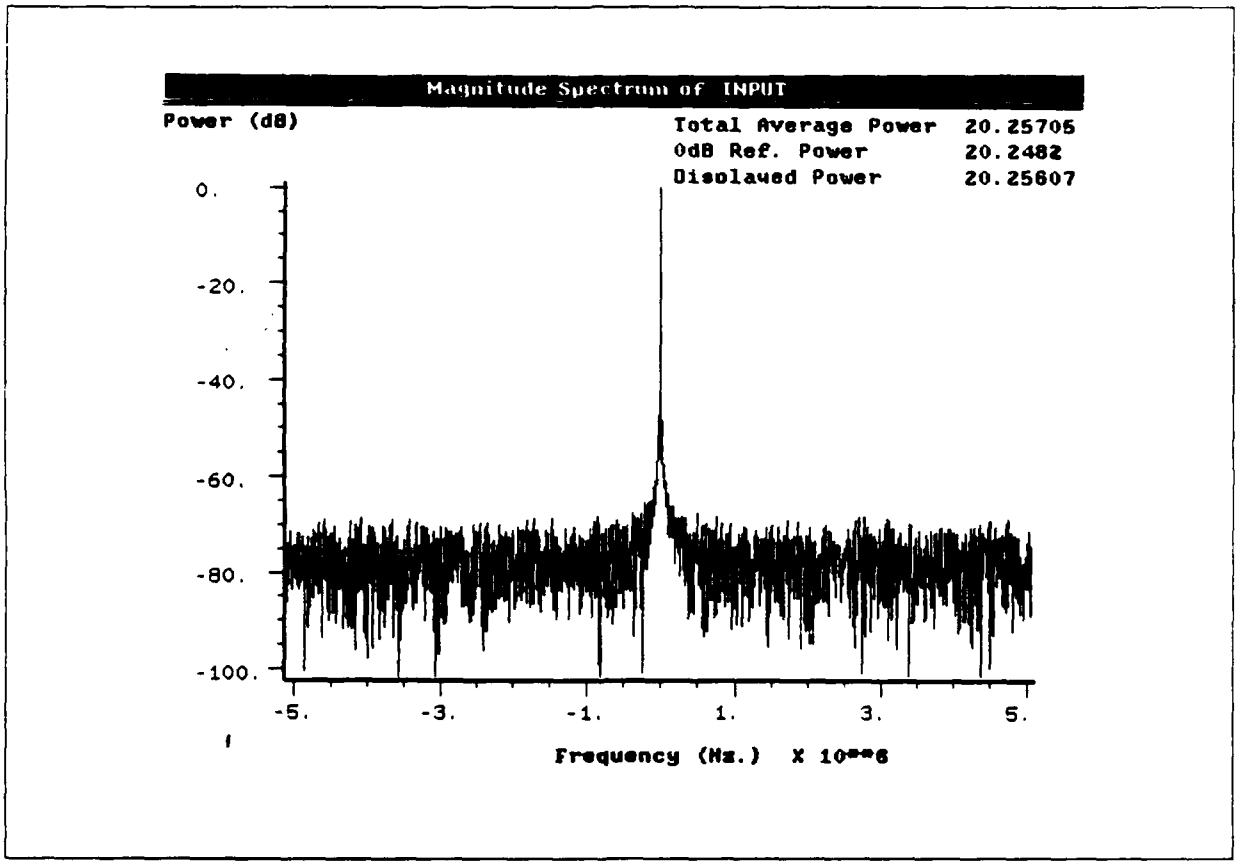

Figure 96. Applebaum Discrete Algorithm Input Spectrum - Sidelobe Canceller

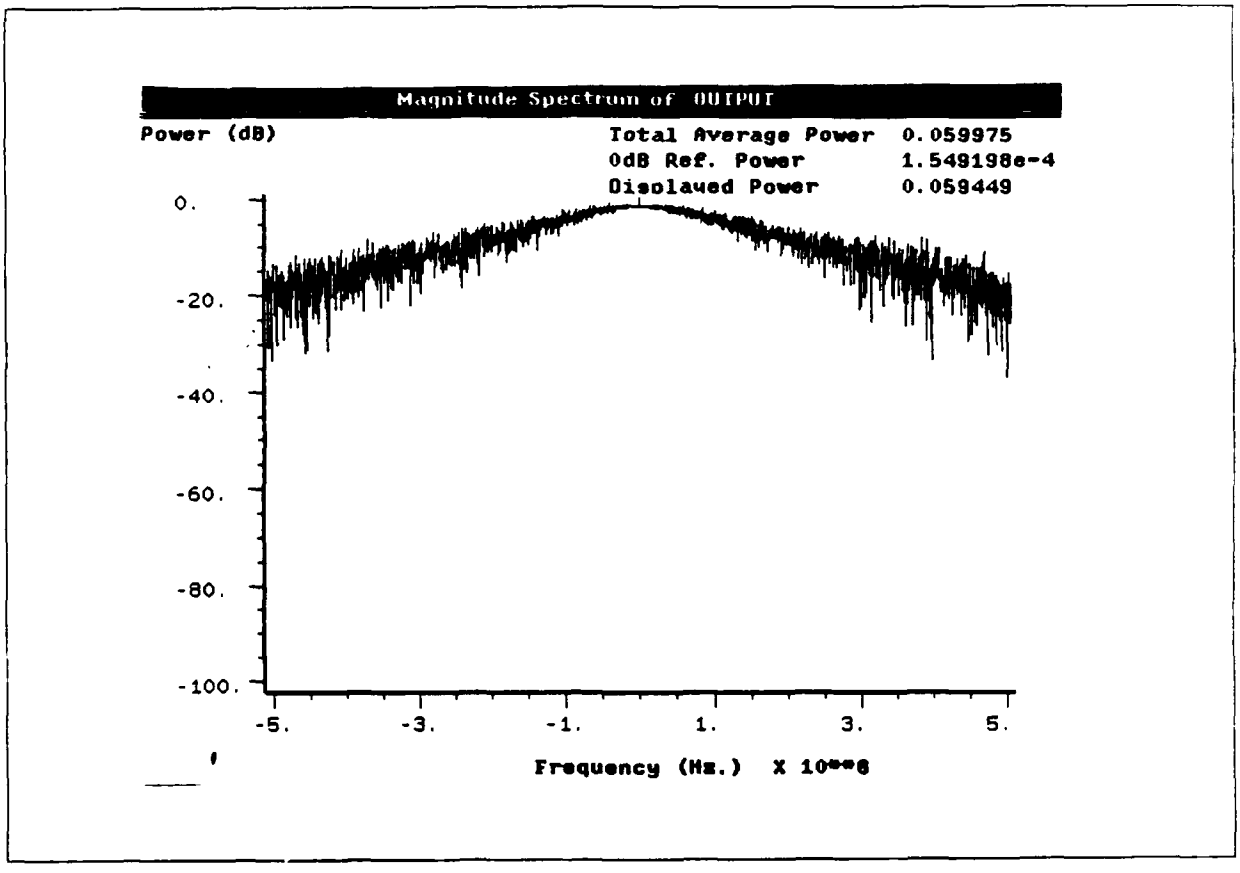

Figure 97. Applebaum Discrete Algorithm Output Spectrum - Sidelobe Canceller

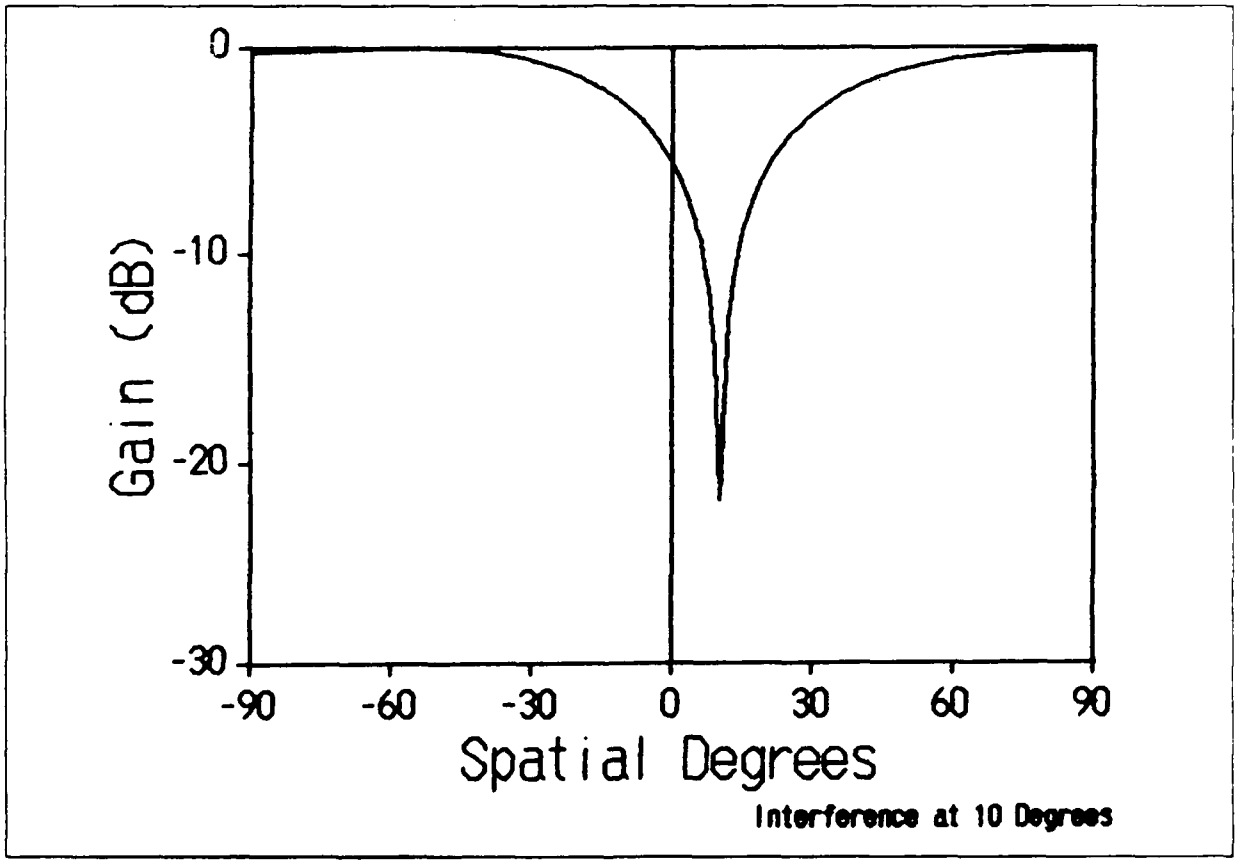

Figure 98. Array Gain with Discrete Algorithm - Jammer at 10 Degrees

A.7 *Four Element Array with Applebaum Loop*

This section includes plots for the four element linear array using the discrete implementation of the Applebaum algorithm. The first simulation put a 20 dB jammer at 15 degrees. The simulation parameters are in Table (19). Figures (99) and (100) contain the input and output magnitude spectrums for the system.

```
STOP-TIME = 3.9999998E-4
DT = 5.0e-8
NOISE CHANNEL 2 = 1
NOISE CHANNEL 1 = 1.0e-4
INPUT LOWPASS FREQ = 5.0e6
GAMMA = (0.001,0.0)
MU = (0.001,0.0)
B2 = (1,0.0)
PHASE SHIFT LOOP4 = 0.0
PHASE SHIFT LOOP3 = 0.0
PHASE SHIFT LOOP2 = 0.0
PHASE SHIFT LOOP1 = 0.0
NOISE ELEMENT 4 PHASE (RADIANS) = 2.439312
NOISE ELEMENT 3 PHASE (RADIANS) = 1.626208
NOISE ELEMENT 2 PHASE (RADIANS) = 0.813104
JAMMER ELEMENT 4 PHASE (RADIANS) = 2.439312
JAMMER ELEMENT 3 PHASE (RADIANS) = 1.626208
JAMMER ELEMENT 2 PHASE (RADIANS) = 0.813104
SIGNAL ELEMENT 4 PHASE (RADIANS) = 0.0
SIGNAL ELEMENT 3 PHASE (RADIANS) = 0.0
SIGNAL ELEMENT 2 PHASE (RADIANS) = 0.0
NOISE POWER = 1.0E-3
JAMMER AMPLITUDE = (4.5,0.0)
SIGNAL AMPLITUDE = (0.001,0.0)
JAMMER FREQUENCY = 50
SIGNAL FREQUENCY = 50
```

Table 19. Applebaum Four Element System with One Jammer Parameters

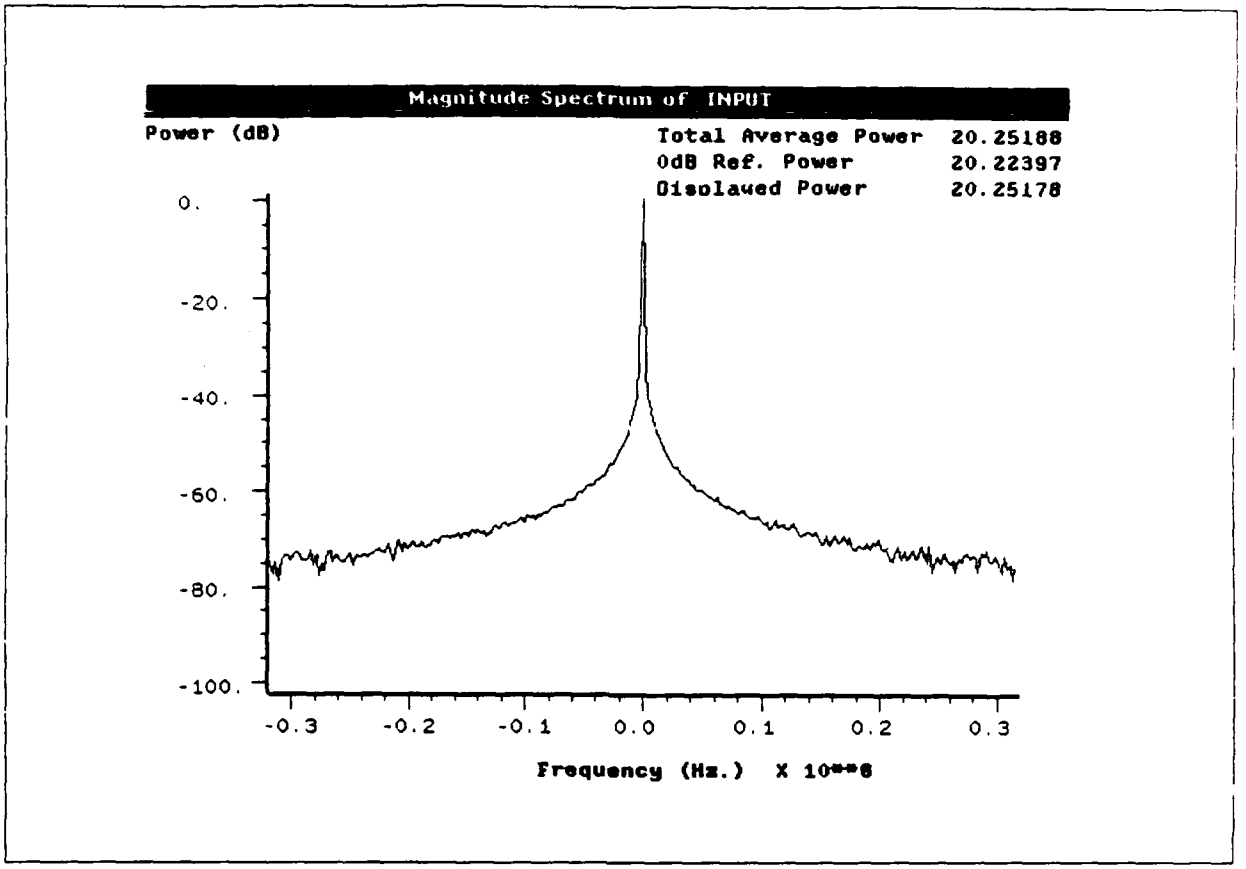

Figure 99. Applebaum Four Element System Input Spectrum - One Jammer

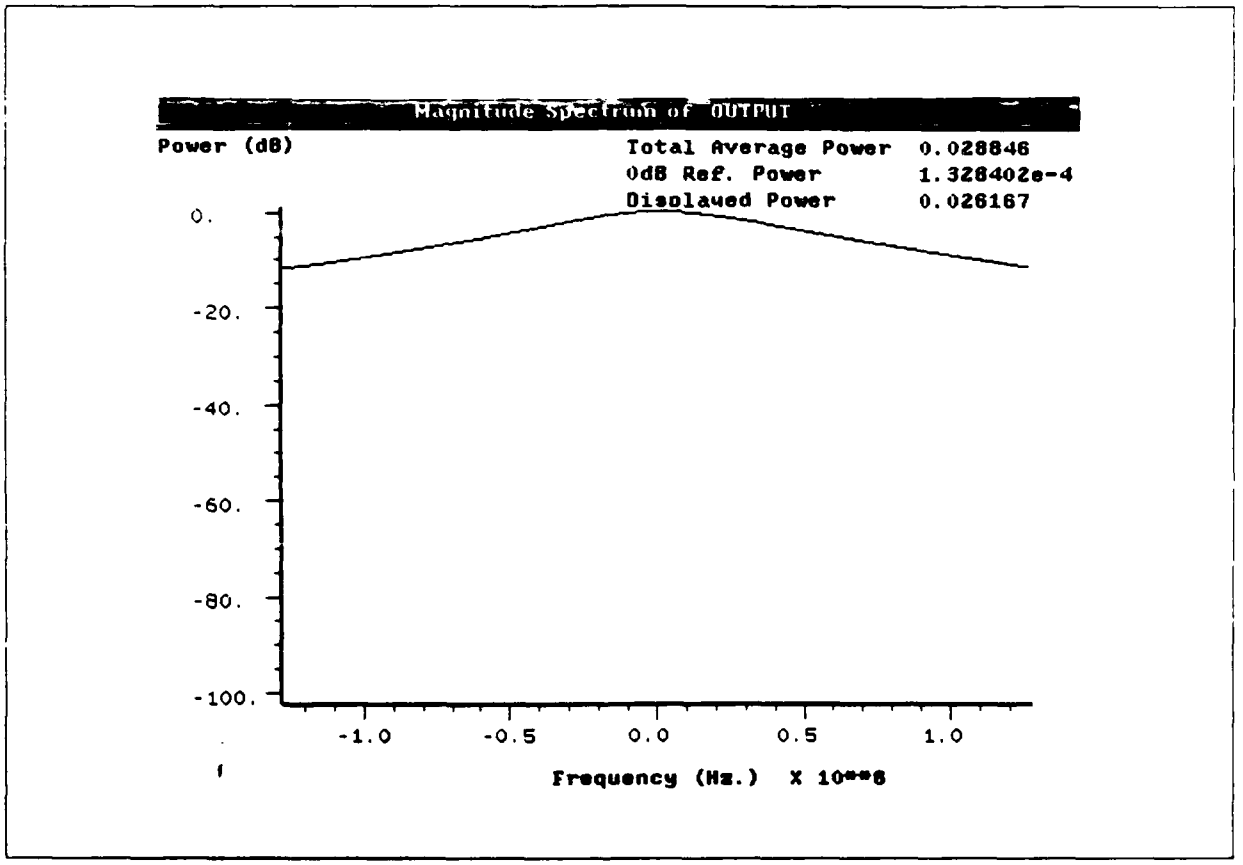

Figure 100. Applebaum Four Element System Output Spectrum - One Jammer

The next simulation added a second 20 dB jammer at -30 degrees. The discrete Applebaum algorithm was again used with a linear four element array. The following table shows the simulation parameters. The next two figures show the input and output magnitude spectrums.

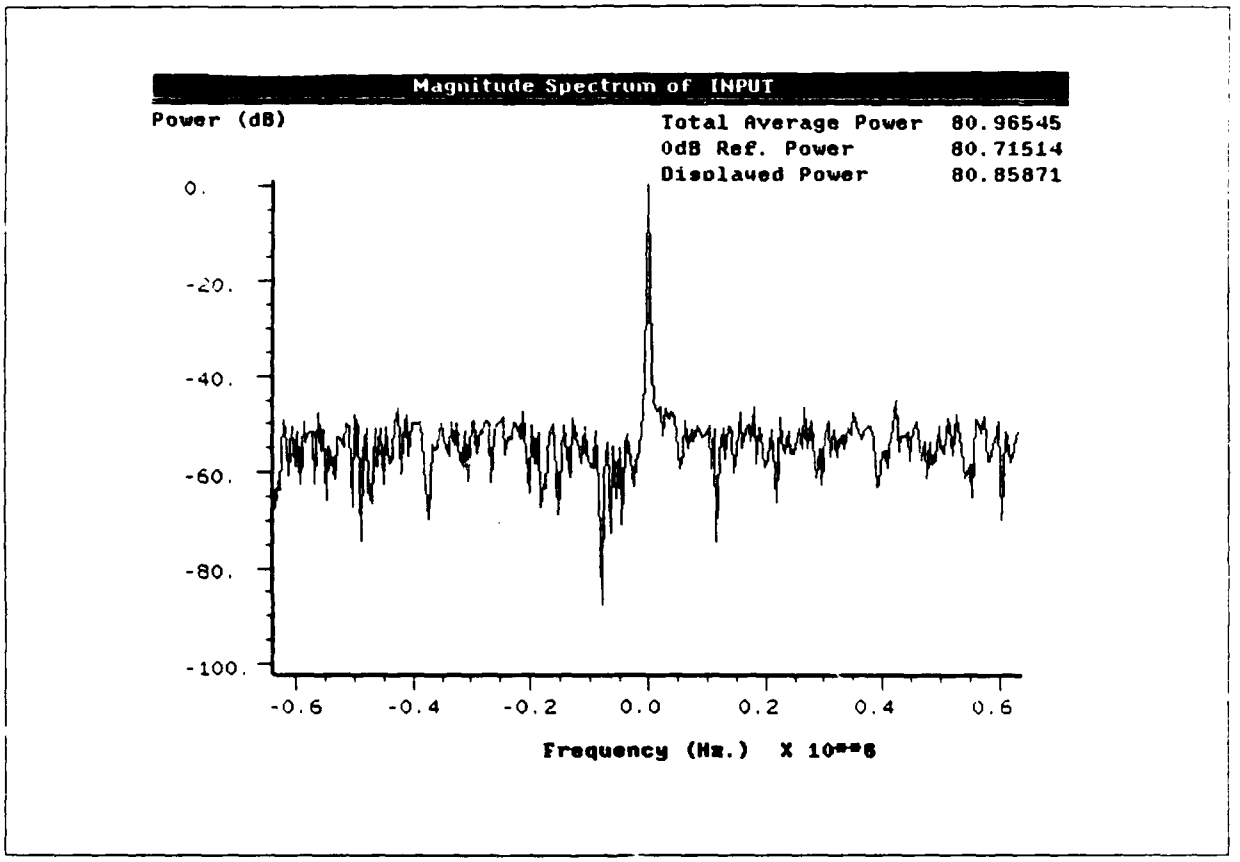

Figure 101. Applebaum Four Element System Input Spectrum - Two Jammers

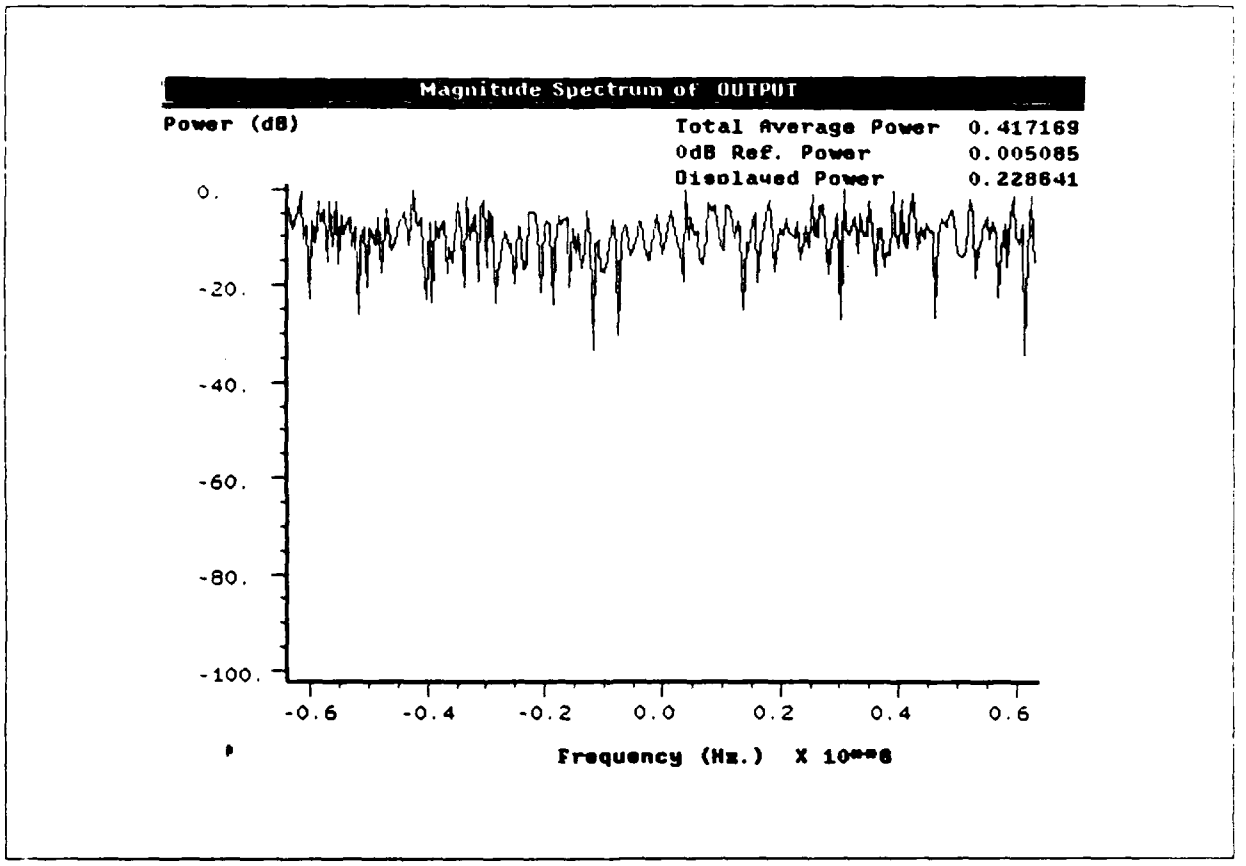

Figure 102. Applebaum Four Element System Output Spectrum - Two Jammers

```
STOP-TIME = 3.9999998E-4
DT = 5.0e-8
NOISE CHANNEL 2 = 1
NOISE CHANNEL 1 = 1
INPUT LOWPASS FREQ = 5.0e6
GAMMA = (0.001,0.0)
MU = (0.001,0.0)
B2 = (1,0.0)
PHASE SHIFT LOOP4 = 0.0
PHASE SHIFT LOOP3 = 0.0
PHASE SHIFT LOOP2 = 0.0
PHASE SHIFT LOOP1 = 0.0
NOISE ELEMENT 4 PHASE (RADIANS) = 2.439312
NOISE ELEMENT 3 PHASE (RADIANS) = 1.626208
NOISE ELEMENT 2 PHASE (RADIANS) = 0.813104
JAMMER ELEMENT 4 PHASE (RADIANS) = 2.439312
JAMMER ELEMENT 3 PHASE (RADIANS) = 1.626208
JAMMER ELEMENT 2 PHASE (RADIANS) = 0.813104
SIGNAL ELEMENT 4 PHASE (RADIANS) = -4.71238
SIGNAL ELEMENT 3 PHASE (RADIANS) = -3.14159
SIGNAL ELEMENT 2 PHASE (RADIANS) = -1.57079
NOISE POWER = 0.1
JAMMER AMPLITUDE = (4.5,0.0)
SIGNAL AMPLITUDE = (4.5,0.0)
JAMMER FREQUENCY = 50
SIGNAL FREQUENCY = 50
```

Table 20. Applebaum Four Element System with Two Jammers Parameters

Appendix B. *BOSS Write to File Module*

BOSS produces unique output files for each parameter monitored during simulation. If two parameters are to be written to a file for observation at a later time, the user must create a module to write the parameters to a file. This was just the case during this study. To look at individual weight values at particular instances in time, a BOSS module was created. The FORTRAN code which handled the file opening, file closing, and output format follows.

```
C    **** BOSS PRIMITIVE *********
C
C    THE FOLLOWING SUBROUTINE IS USED TO WRITE
C    WEIGHT DATA TO A FILE THAT IS EXTERNAL TO
C    BOSS.  THIS FILE WILL KEEP TRACK OF THE
C    TIME HISTORY OF WEIGHT ADAPTATION.
C    ***************************************************
C    AUTHOR:  JOE H. SRUBAR
C    DATE CREATED:  1 AUGUST 1989
C    VARIABLES
C    ---------
C    IS_OPEN  - LOGICAL TO DETERMINE IF FILE IS OPEN
C    LUN      - UNIT NUMBER OF FILE
C    FILENAME - NAME OF FILE TO WRITE INFO TO
C    SAMPLE   - REAL THAT IS TIME OF SAMPLE
C    WEIGHT   - REAL THAT IS THE WEIGHT VALUE AT TIME SAMPLE

      SUBROUTINE SIG_WRITE(IS_OPEN,LUN,FILENAME,SAMPLE,WEIGHT)
      IMPLICIT NONE
      LOGICAL*1 IS_OPEN
      INTEGER LUN
      INTEGER PULSE_CTR
      CHARACTER*(*) FILENAME
      REAL SAMPLE
      REAL WEIGHT
C
C    OPEN THE FILE IF IT HAS NOT ALREADY BEEN OPENED
C
      IF ( .NOT. IS_OPEN) THEN
         CALL LIB$GET_LUN(LUN)
         OPEN (LUN,FILE=FILENAME,STATUS='NEW')
         REWIND(LUN)
         IS_OPEN = .TRUE.
      ENDIF
C
C    WRITE WEIGHT AND TIME DATA TO FILE
C
      WRITE(LUN,*) SAMPLE,WEIGHT
      RETURN
      END
```

Bibliography

1. Applebaum, Sidney P. "Adaptive Arrays," IEEE Transactions on Antennas and Propagation, AP-24, 5, 585-598 (September 1976).
2. Bar-ness, Y. and Pickholtz, R. "Tutorial: Adaptive Array Processing and Interference Cancellation," MILCOM 1988.
3. Block Oriented Systems Simulator, BOSS, User's Guide, COMDISCO Corporation, January 1989.
4. Compton, R. T. Adaptive Antennas, Concepts and Performance, Englewood Cliffs, New Jersey: Prentice Hall, 1988.
5. Gabriel, Willaim F. "Adaptive Arrays - An Introduction," Proceedings of the IEEE, 64, Number 2, 239–272, (February 1976).
6. Griffiths, L.J. "A Simple Adaptive Algorithm for Real-Time Processing in Antenna Arrays," Proceedings of the IEEE, 57, Number 10, 1696, (October 1969).
7. Pipes, Louis A. and Havanessian, Shahen A. Matrix Computer Methods in Engineering, Huntington, New York: Robert E. Krieger, 1977.
8. Riski, Capt William A. Effects of Adaptive Antenna Arrays on Broadband Signals. MS Thesis AFIT/GE/EE/80–4. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, June 1980 (AD-A085 708).
9. Widrow, Bernard and Stearns, Samuel Adaptive Signal Processing, Englewood Cliffs, New Jersey: Prentice Hall, 1985.
10. Widrow, B. and others "Adaptive Antenna Systems," Proceedings of the IEEE, 55, No. 12, 2143-2159 (December 1967).
11. Widrow, Bernard and others "The Complex LMS Algorithm," Proceedings of the IEEE, 63, No.4, 719-720 (April 1975).

Vita

Captain Joe H. Srubar [redacted] After entering the United States Air Force in September 1972, he was eventually trained as an Avionics Fire Control Technician. In 1982 he attended Texas A & M University under the Airman Education and Commissioning Program and received a Bachelor of Science degree in Electrical Engineering in December 1984. Upon completion of Officer's Training School in April 1985, he received his commission and was assigned to the Navigation Branch of the Aeronautical Systems Division at Wright Patterson Air Force Base. He entered the School of Engineering, Air Force Institute of Technology, in May 1988.

[redacted]

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GE/ENG/89D-51	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB OH 45433-6583	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Communication Technology Branch 8b. OFFICE SYMBOL (If applicable) WRDC/AAAI-2	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB OH 45433-6583	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
IMPLEMENTATION OF ADAPTIVE ARRAYS IN THE BLOCK ORIENTED SYSTEMS SIMULATOR

12. PERSONAL AUTHOR(S)
Joe H. Gruhar, B. S., Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS Thesis	FROM _____ TO _____		

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Adaptive Array Algorithms Digital Simulation
06	04		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Advisor:
David M. Norman, Lt Col, USAF
Assistant Professor

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
David M. Norman, Assitant Professor	513-255-3708	ENG

DD Form 1473, JUN 86 *Previous editions are obsolete.*

This study investigated the feasability of implementing adaptive array algorithms in a simulation program called the Block Oriented Systems Simulator (BOSS). Two algorithms in particular were used: the least mean square (LMS) algorithm in real and complex form originally developed by Widrow, et. al. and the Applebaum algorithm. Two arrays were simulated with simple isotropic radiators used as the array elements. The first array used two elements spaced one half wavelength apart. The second array used four elements with variable geometry. The approach taken was to first implement the algorithms and arrays in the simulation language. Testing the algorithms with simple inputs and checking the algorithms ability to track these inputs was the second step. The last step was to test the algorithms with the arrays.

The results of the simulations showed the LMS algorithms implemented in BOSS were able to track DC signals along with sinusoidal inputs. The BOSS LMS implementation was also able to reject jammer signals by changing the gain and phase of the array elements. The amount of signal to jammer ratio was only a function of the gain limits put on the LMS loops. An analog implementation of the Applebaum algorithm did not perform as well. When used in a sidelobe canceller circuit with one adaptive weight, the complex weight would initially converge to a theoretical optimum weight but eventually would diverge. This problem was corrected with a discrete implementation of the algorithm. The discrete implementation provided weight convergence with nulls being placed at a jammer's spatial location. The null depth exactly matched the jammer power.

The simulations from this study show that adaptive array algorithms can be implemented and simulated with BOSS. The BOSS simulation environment proved to be an excellent tool for prototyping systems along with a providing a rich selection of post simulation output products.